BOUQUET: A SATELLITE CONSTELLATION VISUALIZATION PROGRAM FOR WALKERS AND LATTICE FLOWER CONSTELLATIONS

A Thesis

by

MANDAKH ENKH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2011

Major Subject: Aerospace Engineering

Bouquet: A Satellite Constellation Visualization Program

for Walkers and Lattice Flower Constellations

BOUQUET: A SATELLITE CONSTELLATION VISUALIZATION PROGRAM FOR WALKERS AND LATTICE FLOWER CONSTELLATIONS

A Thesis

by

MANDAKH ENKH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Daniele Mortari
Committee Members,	John Hurtado
	John Junkins
	J. Maurice Rojas
Head of Department,	Dimitris Lagoudas

August 2011

Major Subject: Aerospace Engineering

ABSTRACT

Bouquet: A Satellite Constellation Visualization Program for Walkers and Lattice Flower Constellations. (August 2011)

Mandakh Enkh, B.S., Texas A&M University

Chair of Advisory Committee: Dr. Daniele Mortari

The development of the Flower Constellation theory offers an expanded framework to utilize constellations of satellites for tangible interests. To realize the full potential of this theory, the beta version of Bouquet was developed as a practical computer application that visualizes and edits Flower Constellations in a user-friendly manner. Programmed using C++ and OpenGL within the Qt software development environment for use on Windows systems, this initial version of Bouquet is capable of visualizing numerous user defined satellites in both 3D and 2D, and plot trajectories corresponding to arbitrary coordinate frames. The ultimate goal of Bouquet is to provide a viable open source alternative to commercial satellite orbit analysis programs. As such, the coding of Bouquet puts heavy emphasis on flexibility, upgradability and methods to provide continued support through open source collaboration.

ACKNOWLEDGEMENTS

I would like to extend a big thanks to Dr. Mortari for his continued encouragement, support, pushing me to do my best, and most importantly for giving me the opportunity to take on this project.

Thanks also to Drs. Martin Avendaño and Christian Bruccoleri for their expertise, insight, and laying the foundations of the project that has helped me learn and move forward.

NOMENCLATURE

Ω	Right ascension of ascending node
ω	Argument of perigee
ω_d	Angular velocity of a rotating frame
2D	Two-dimensional
3D	Three-dimensional
a	Semi-major axis
AGI	Analytical Graphics, Inc.
API	Application programming interface
c1	Starting column, scalar
c2	Ending column, scalar
e	Eccentricity
E_d	Original Flower Constellation integer parameter
E_n	Original Flower Constellation integer parameter
f	Phasing of satellites in Walker Constellation
F_d	Original Flower Constellation integer parameter
F_h	Original Flower Constellation integer parameter
F_n	Original Flower Constellation integer parameter
FC	Flower Constellation
FCVAT	Flower Constellation visualization and analysis tool
GMAT	General mission analysis tool

GUI	Graphical user interface
H	Lower Hermite normal form of lattice matrix
$\widetilde{H}$	Upper Hermite normal form of lattice matrix
HFC	Harmonic Flower Constellation
i	Inclination
J_2	Orbit perturbation effect from Earth oblateness
LFC	Lattice Flower Constellation
M	Mean anomaly
MFC	Microsoft foundation classes
MSDN	Microsoft developer network
n	Mean motion
N_c	Configuration number
N_c'	Dual configuration number
N_d	Number of planetary rotation periods, integer
N_m	Number of different mean anomalies
N_o	Number of orbits
N_p	Number of satellite orbit periods, integer
N_s	Total number of satellites
N_{sm}	Number of orbits containing a given mean anomaly
N_{so}	Number of satellites per orbit
OFC	Original Flower Constellation
OOP	Object oriented programming

OS	Operating system
p	Number of orbits in Walker Constellation
r1	Starting row, scalar
r2	Ending row, scalar
RAAN	Right ascension of ascending node
RAM	Random access memory
SDK	Software development kit
STK	Satellite tool kit
t	Total number of satellites in Walker Constellation
T	Satellite orbit period
T_d	Planet rotation period
T_{rep}	Relative compatible orbit period
VS	Microsoft visual studio
WPF	Windows presentation foundation

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION: BOUQUET

The purpose of Bouquet is to serve as a central tool for future engineers and developers to analyze and work with the Flower Constellation (FC) concept. Bouquet aims to encompass all the necessary code to set up the constellation, simulate orbital paths with user chosen propagators, and particularly provide the framework to be able to optimize the constellation such that it meets specific performance goals. A framework for optimization naturally requires flexibility in parameters and systems, and as such flexibility forms one of the fundamental design goals.

Bouquet's purpose is defined by four pillars of operation.

1) *Simulation*. Be able to simulate any non-hyperbolic orbit of numerous satellites given initial conditions, a defined planet, and a two-body orbital propagation method over an arbitrary length of time.
2) *Visualization*. Be able to clearly display the satellites, planet and orbital paths – inertial and relative – in both 3D and 2D (with given projection method).
3) *Editing*. Be flexible enough to handle and modify orbital parameters, number of satellites, constellation types, relative coordinate frames, calculation methods, and planetary characteristics as necessary.
4) *Optimization*. Be able to automatically modify constellation parameters and generate the best structure for a given purpose.

This thesis follows the style of *AIAA Journal*.

2. FLOWER CONSTELLATIONS

2.1 Overview

An introduction into Bouquet cannot start until a firm understanding of Flower Constellations is established. FCs are a specific type of satellite constellation based on the compatible orbits idea (i.e., resonant orbits or repeating ground track orbits).

The greatest commercial purpose of satellites, and satellite constellations, is to interact with the Earth and its surface through communications and sensors. So the satellites' ground track – directly related to their ability to observe and interact with a given point on the surface – is of special importance. The nature of the ground track is determined by the combination of the satellites' orbit and the rotation of the planet. Compatible orbits are a specialization of this combination such that the relation

$$N_p T = N_d T_d = T_{\text{rep}} \; where \; N_p \; and \; N_d \; are \; integers \tag{1}$$

is satisfied. Since N_p and N_d are integers, the ground track is guaranteed to repeat itself in T_{rep}time, forming the basis of compatible orbits. This ground track is defined by the relative orbit – the satellite path as seen from the planet surface – so that the relative orbit forms a closed loop. Figure 2-1 illustrates the relative and inertial trajectories.

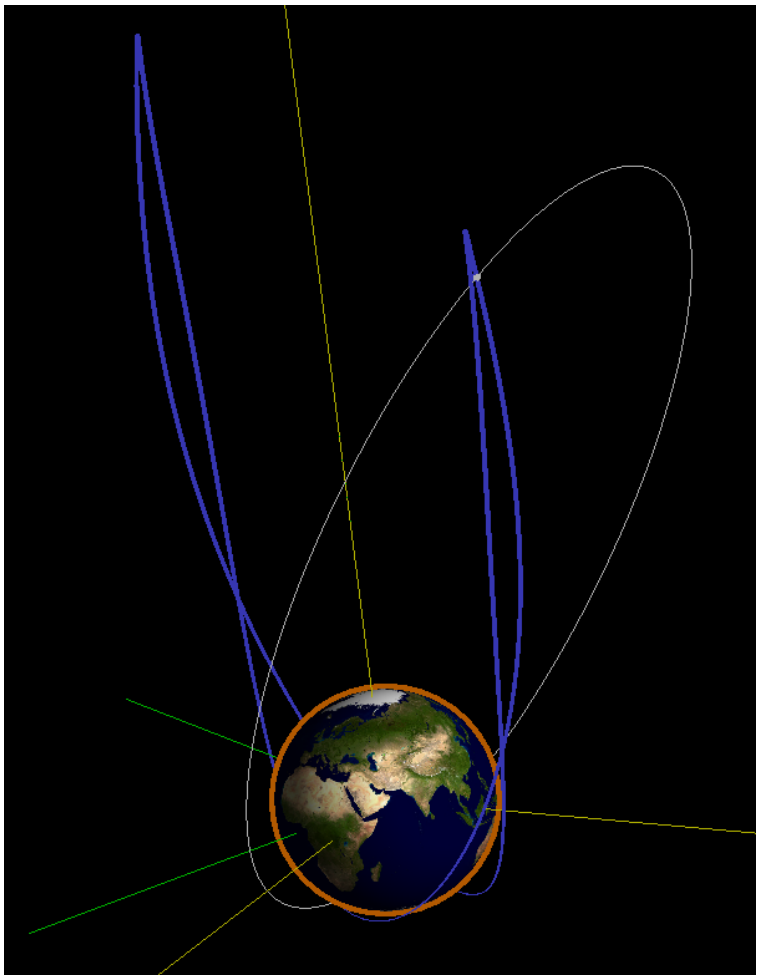

Figure 2-1. The relative and inertial trajectories of a Molniya orbit.

Coincidentally, these relative compatible orbits often form intricate patterns resembling flower petals, hence the name Flower Constellation. Originally conceptualized in 1967 by Dr. Luigi Broglio from University of Rome La Sapienza as a four satellite constellation called “Sistema Quadrifoglio” (four-leaf system), it has been expanded and developed into its present form by Dr. Mortari and his PhD students Matthew P. Wilkins and Christian Bruccoleri [1].

There are two ways to build FCs. One is to array the satellites – each with the

same orbit shape, inclination and argument of perigee – to varying RAAN (Figure 2-2) such that the mean anomalies are as described in Equation 2.

$$M_2(0) = M_1(0) + n\frac{\Omega_1 - \Omega_2}{\omega_d} \text{ (ref. [2])} \tag{2}$$

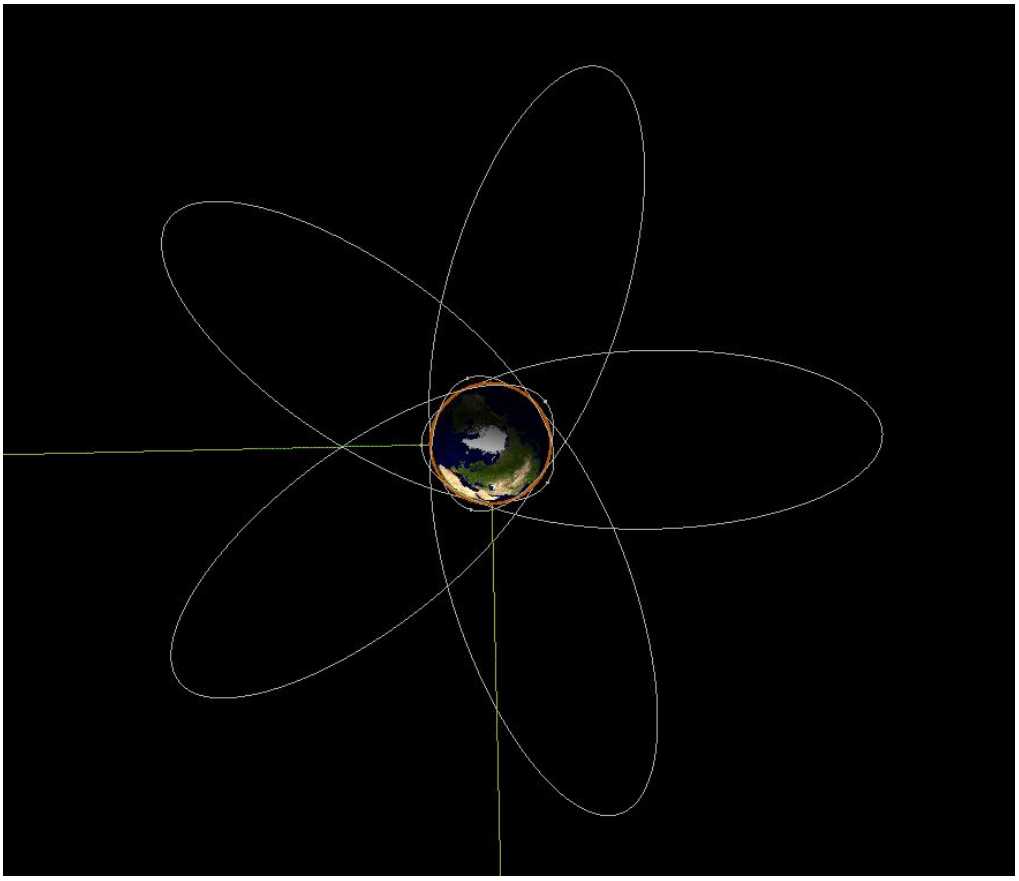

Figure 2-2. Constellation with RAAN varied inertial orbits.

The second method is to array the satellites into a single inertial orbit at specific positions within the orbit such that all satellites have the same relative orbit, forming a

closed FC (Figure 2-3). Mathematically there are only N_d number of such admissible positions for each inertial orbit [2]. From Equation 2 we get

$$M_{k+1}(0) = M_k(0) + n\frac{2\pi}{\omega_d} \; where \; 2\pi = \Omega_1 - \Omega_2 \tag{3}$$

$$Since \; n = \frac{2\pi}{T} \; and \; \omega_d = \frac{2\pi}{T_d}$$

$$M_{k+1}(0) = M_k(0) + 2\pi\frac{T_d}{T}$$

$$M_{k+1}(0) = M_k(0) + 2\pi\frac{N_p}{N_d} \; so \; that \; \gcd(N_p, N_d) = 1 \; (\text{ref. } [2]) \tag{4}$$

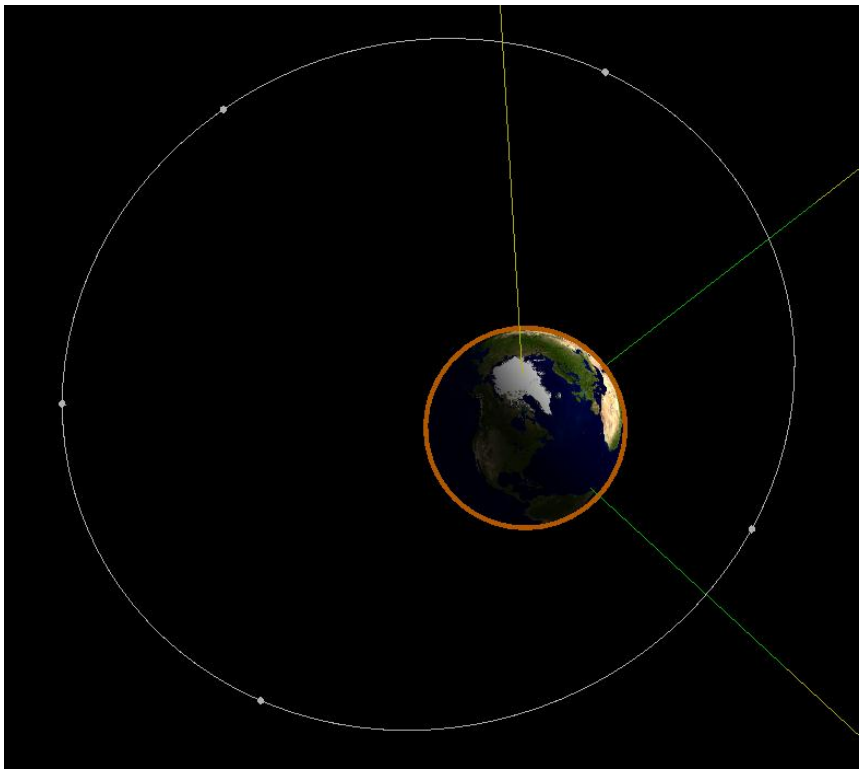

Figure 2-3. Constellation with a single inertial orbit.

An infinite number of unique FCs can be created using a combination of these two methods to array the satellites. The advantages associated with FCs can be utilized to create time-uniform and space-uniform constellations that can directly benefit applications in the industry. Time-uniform constellations can be used to pass over surface points at specific time intervals for reconnaissance, sensing and communications purposes. Space-uniform constellations can be made to regulate distances between the satellites and uniformly cover large surface areas.

2.2 Walker Constellations

The Walker Constellation theory predates the FC theory and encompasses a narrower design. Walker Constellations have identical circular orbits uniformly distributed over Ω and uniformly distributed satellites over M in each orbit [3]. Figure 2-4 illustrates a Walker Constellation.

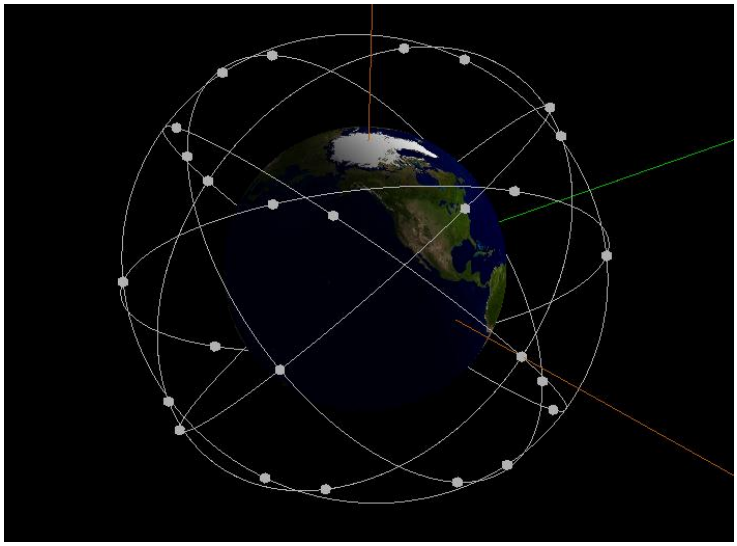

Figure 2-4. Walker Constellation. $a = 11000\ km, e = 0, i = 45.84\ deg, t = 36, p = 6, f = 0.$

Being equivalent to a subset of FCs, Walkers can be represented by FC theory as outlined in Table 2-1. Table 2-2 looks at the advantages of FCs over the Walker Constellation system. The advantages of two-way orbits, dual compatible orbits and harmonic constellations are of special note.

Table 2-1. Walker representation by FC theory.

Walker	**FC**
i (inclination)	i
t (total number of satellites)	$N_o N_{so}$
p (number of orbits)	N_o
f (phasing of satellites in adjacent orbits)	$N_c \bmod N_o$

Table 2-2. Comparison of Walker and Flower Constellations.

Capabilities	**FC**	**Walker**
Circular orbits	Yes	Yes
Elliptical orbits	Yes	No
Free choice of inclination	Yes	Yes
Choice of revisiting time with multiple satellites	Yes	No
Repeating ground track	Yes	Yes
Two-way orbits	Yes	No
Dual compatible orbits	Yes	No
Multi-stationary orbits	Yes	No
Sun synchronous	Yes	Yes
Arbitrary number of satellites	Yes	No
Harmonic constellations	Yes	No

2.3 Original Flower Constellations (OFC)

The initial approach to FCs was markedly different from its current form (Lattice Flower Constellations) and is referred to as Original Flower Constellations (OFC). All satellites in an OFC are constrained to a single relative trajectory that is closed (i.e.,

compatible orbits are required), which translate into three mathematical conditions:

$$N_p T_p = N_d T_d \text{ where } \gcd(N_p,\ N_d) = 1 \text{ (orbit compatibility)}$$

$$N_p \Omega_i = -N_d M_i \bmod 2\pi \text{ (single relative trajectory)}$$

$$a,\ e,\ i,\ \omega \text{ same for each satellite}$$

To satisfy these conditions, $N_p,\ N_d,\ T_d$ are selected which define the satellite orbital period T_p, in turn resulting in the semi-major axis of each orbit, a. The parameters $N_s,\ e,\ i, \omega$ are selected independently. Finally, the integer parameters $F_n,\ F_d,\ F_h$ are selected which define the $(\Omega_i,\ M_i)$ pairs for each satellite of the constellation through a recursive algorithm [2]. This approach of construction means the OFC is defined by six independent parameters, $(N_p,\ N_d,\ F_n,\ F_d,\ F_h, N_s)$, excluding the orbital parameters.

The $(\Omega_i,\ M_i)$ pairs determine the position of each satellite. A constellation with the maximum number of satellites possible, $\frac{N_d F_d}{G}$ where $G = \gcd(N_d,\ N_p F_n + F_d F_h)$, is called a Harmonic Flower Constellation (HFC). The $(\Omega_i,\ M_i)$ pairs of an HFC can be said to be determined from three invariants:

$$N_o = F_d, \text{ the number of inertial orbits}$$

$$N_{so} = \frac{N_d}{G} \text{ , the number of satellites per orbit}$$

$$N_c = E_n \frac{N_p F_n + F_d F_h}{G} \bmod F_d, \text{ the configuration number} \in [0,\ F_d), \text{ where}$$

$$E_n F_n + E_d F_h = 1 \text{ and } \gcd(N_o,\ N_c,\ N_{so}) = 1$$

These invariants, $N_o,\ N_c,$ and N_{so}, are tangible ideas of the constellation and are useful for characterization.

There are two main problems with the OFC approach:

- Constellation definitions can be equivalent (non-minimal representation).
- Can't prove that definitions cover all possibilities.

To solve these issues, the Lattice Flower Constellation approach was developed.

2.4 Lattice Flower Constellations (LFC)

The current and improved approach to FCs is the Lattice Flower Constellation approach. LFC development can be further divided into 2D and 3D versions, referring to the number of dimensions the defining matrix takes. The 2D LFC uses a 2x2 integer matrix to construct constellations but does not take into account any orbit perturbation calculations. The 3D LFC, using a 3x3 integer matrix, takes into account the perturbation effects, specifically Ω and ω precession resulting from the J_2 effect. As Bouquet doesn't currently handle 3D LFC, this FC introduction will focus on the 2D LFC theory.

The main idea behind the LFC approach is the question "why are the restraint conditions necessary in the OFC approach?" As it turns out, those conditions –

compatible orbits, single relative trajectory, and $\gcd(N_o,\ N_c,\ N_{so}) = 1$ – are not necessary. Thus, LFCs don't have to have closed relative trajectories and may have multiple relative trajectories depending on any rotation frame specified. The definition of the constellation is decoupled from the rotation frame. The LFC approach gives a minimal parameter representation, three, and can be proven to cover all symmetric solutions [4].

For the 2D LFC approach, the positions of each satellite – defined by $(\Omega_i,\ M_i)$ pairs – can be directly obtained from the tangible $N_o,\ N_c,$ and N_{so} parameters (in contrast to the OFC approach where it was the other way). Equation 5 illustrates the lattice matrix corresponding to the $(\Omega_i,\ M_i)$ pairs.

$$L\begin{bmatrix}\Omega\\M\end{bmatrix} \equiv \begin{bmatrix}0\\0\end{bmatrix}\ mod\ 2\pi \text{ where } L = \begin{bmatrix}a & b\\c & d\end{bmatrix} \tag{5}$$

$$H\begin{bmatrix}\Omega\\M\end{bmatrix} \equiv \begin{bmatrix}0\\0\end{bmatrix}\ mod\ 2\pi \text{ where } H = \begin{bmatrix}N_o & 0\\N_c & N_{so}\end{bmatrix} \tag{6}$$

$$\widetilde{H}\begin{bmatrix}\Omega\\M\end{bmatrix} \equiv \begin{bmatrix}0\\0\end{bmatrix}\ mod\ 2\pi \text{ where } \widetilde{H} = \begin{bmatrix}N_{sm} & N_c'\\0 & N_m\end{bmatrix} \tag{7}$$

The lattice matrix L can be decomposed into equivalent H and $\widetilde{H}$ forms by performing a sequence of integer row operations, resulting in the lower and upper Hermite normal forms respectively [4]. The lower and upper Hermite normal forms are minimal representations and correspond to a unique FC configuration. The equivalency is demonstrated in Figure 2-5.

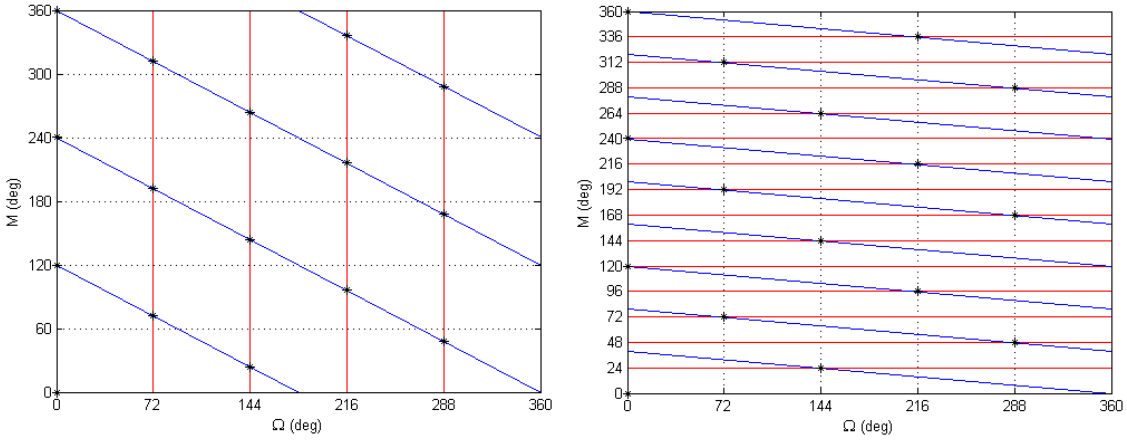

Figure 2-5. Lower and upper Hermite normal form equivalency [4]. $H = \begin{bmatrix} 5 & 0 \\ 2 & 3 \end{bmatrix}$ and $\tilde{H} = \begin{bmatrix} 1 & 9 \\ 0 & 15 \end{bmatrix}$.

The lower Hermite normal form, as shown in Equation 6, represents the N_o, N_c, and N_{so} parameters and is the most useful form to construct 2D LFCs.

3. SIMILAR APPLICATIONS IN THE FIELD

3.1 Flower Constellation Visualization and Analysis Tool (FCVAT) by Dr. Christian Bruccoleri

A prototype application, programmed by Dr. Mortari's former graduate student Dr. Christian Bruccoleri, was developed several years ago to demonstrate the concept of FCs (Figure 3-1). Written in Java and featuring 3D display capabilities, its main goal is to visualize and demonstrate an arbitrary FC. The user is allowed to choose the number of satellites and other basic parameters of an FC, e.g., number of petals.

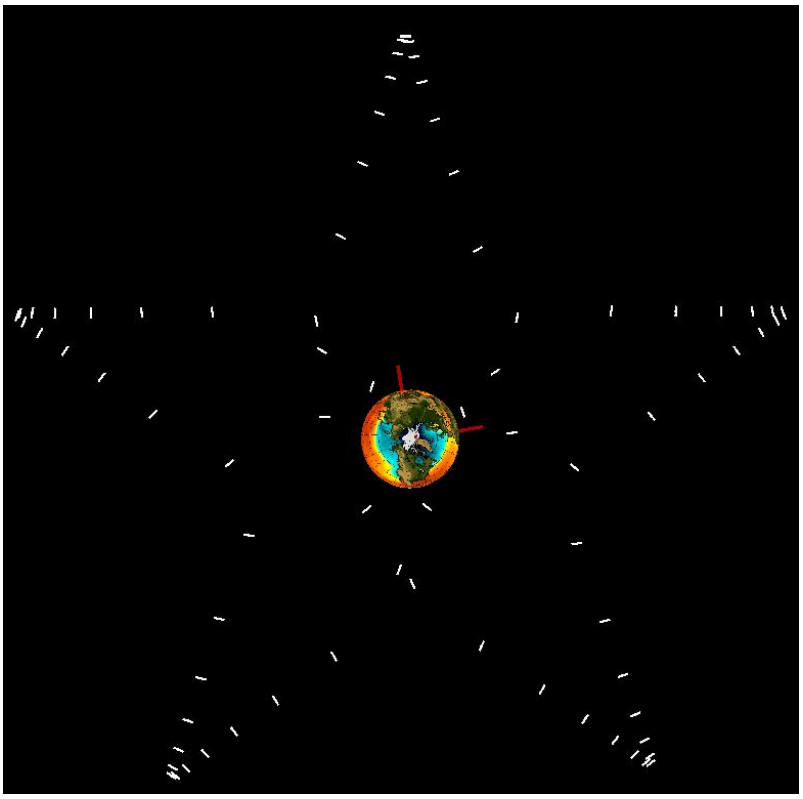

Figure 3-1. The 3D display of FCVAT.

Of existing applications, this prototype is the most similar to Bouquet by virtue of its focus on FC simulation, but only focuses on the old OFC approach. Bouquet adds 2D LFC capability and further improves on the prototype in a number of ways as illustrated in Table 3-1.

Table 3-1. Comparison of FCVAT and Bouquet capabilities

Capabilities (Detailed)	**FCVAT**	**Bouquet**
3D visualization of FC	Yes	Yes
2D projection visualization of FC	No	Yes
Draw relative trajectories	Yes	Yes
Draw inertial trajectories	Yes	Yes
Basic FC input (e.g., orbital parameters, number of petals)	Yes	Yes
Advanced FC input (e.g., lattice construction, model switching)	No	Yes
Planet interchangeability	No	Yes
Draw coordinate frames	No	Yes
Arbitrary relative coordinate frames	No	Yes
Arbitrary relative trajectories	No	Yes
Incompatible orbits	No	Yes
Sun and day/night simulation	No	Yes
Planet axial rotation simulation	No	Yes
Basic time controls (e.g., stop, play)	Yes	Yes
Advanced time controls (e.g., skip, initial time, final time)	No	Yes
Preferences controls (e.g., adjust colors, visibility, units)	No	Yes
Texture rendering	Yes	Yes
OFC	Yes	Yes
2D LFC	No	Yes

The prototype is wholly focused on providing a 3D simulation of FCs without any flexible architecture considerations for optimization. Bouquet improves on this by adding broad flexibility and editing capabilities along with other user oriented features.

3.2 Ikebana by Dr. Martin Avendaño

Ikebana (named after the Japanese art of flower arrangement) is another

important FC demonstrator programmed by Dr. Mortari's colleague, Dr. Martin Avendaño (Figure 3-2). Ikebana is an extremely lightweight application designed solely to illustrate FCs in a minimal 3D display. Using ingeniously mathematical routines, it manually displays the necessary 3D objects without even using APIs such as OpenGL or DirectX.

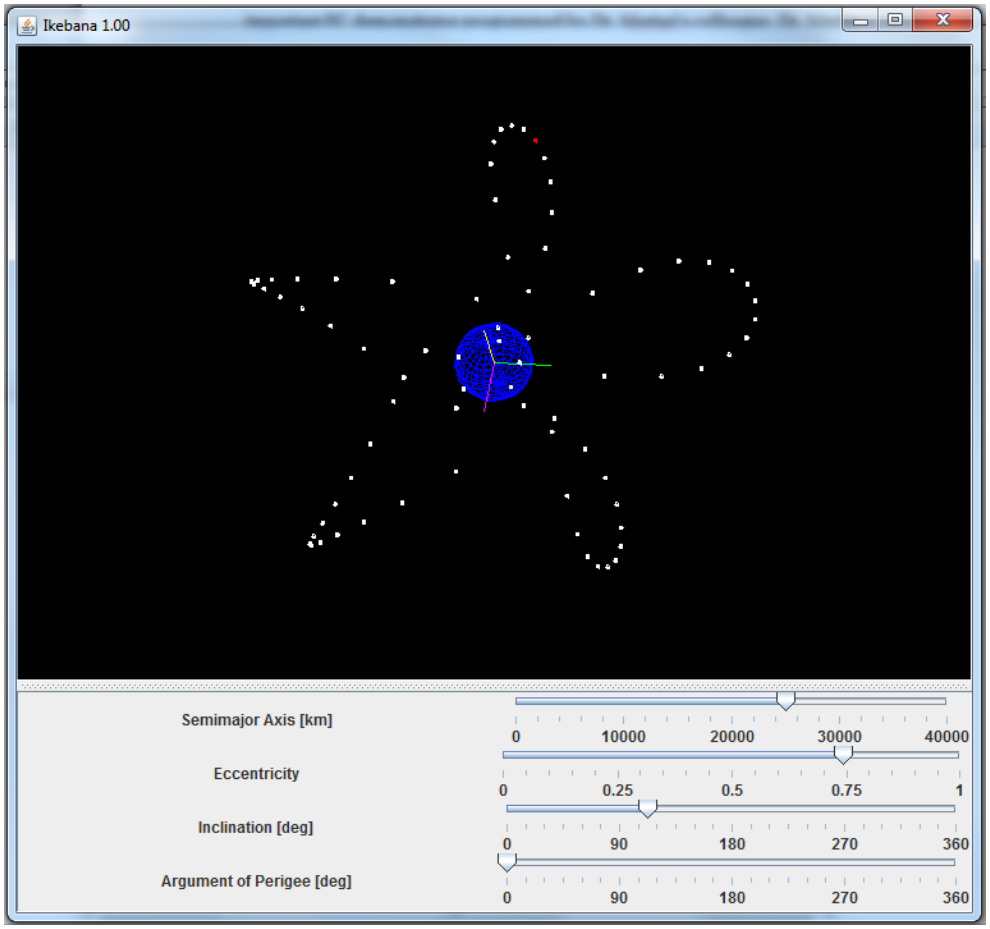

Figure 3-2. Ikebana GUI.

The simulated FC is hard-coded except for slider based inputs to change its

orbital parameters. The goal of Ikebana is not to be a full-fledged application to be used in the design of FCs and thus does not include capabilities of such (Table 3-2).

Table 3-2. Comparison of Ikebana and Bouquet capabilities.

Capabilities (Detailed)	Ikebana	Bouquet
3D visualization of FC	Yes	Yes
2D projection visualization of FC	No	Yes
Draw relative trajectories	No	Yes
Draw inertial trajectories	No	Yes
Basic FC input (e.g., orbital parameters, number of petals)	Yes[1]	Yes
Advanced FC input (e.g., lattice construction, model switching)	No	Yes
Planet interchangeability	No	Yes
Draw coordinate frames	No	Yes
Arbitrary relative coordinate frames	No	Yes
Arbitrary relative trajectories	No	Yes
Incompatible orbits	No	Yes
Sun and day/night simulation	No	Yes
Planet axial rotation simulation	No	Yes
Basic time controls (e.g., stop, play)	No	Yes
Advanced time controls (e.g., skip, initial time, final time)	No	Yes
Preferences controls (e.g., adjust colors, visibility, units)	No	Yes
Texture rendering	No	Yes

[1] Limited input. Does not allow input of number of satellites and number of petals.

Despite its minimal focus, a couple of important Bouquet routines were based on the programming of Ikebana. These include the rotation of the 3D display using the mouse and the construction method of basic FC orbital parameters.

3.3 STK: Satellite Tool Kit

Satellite Tool Kit (STK), a proprietary application developed and supported by Analytical Graphics, Inc. (AGI), is a powerful user tested application and the most commercially successful satellite analysis tool on the market today (Figure 3-3). During its more than three decades of development, STK has gathered an impressive list of

features and offer comprehensive satellite analysis and design tools ranging from the minute details such as solar pressure to major considerations like attitude control and collision detection. The domain of STK capability is not limited to satellites, also offering the ability to analyze many different objects within the broader aerospace and defense domain such as airplanes, radar installations and naval ships [5].

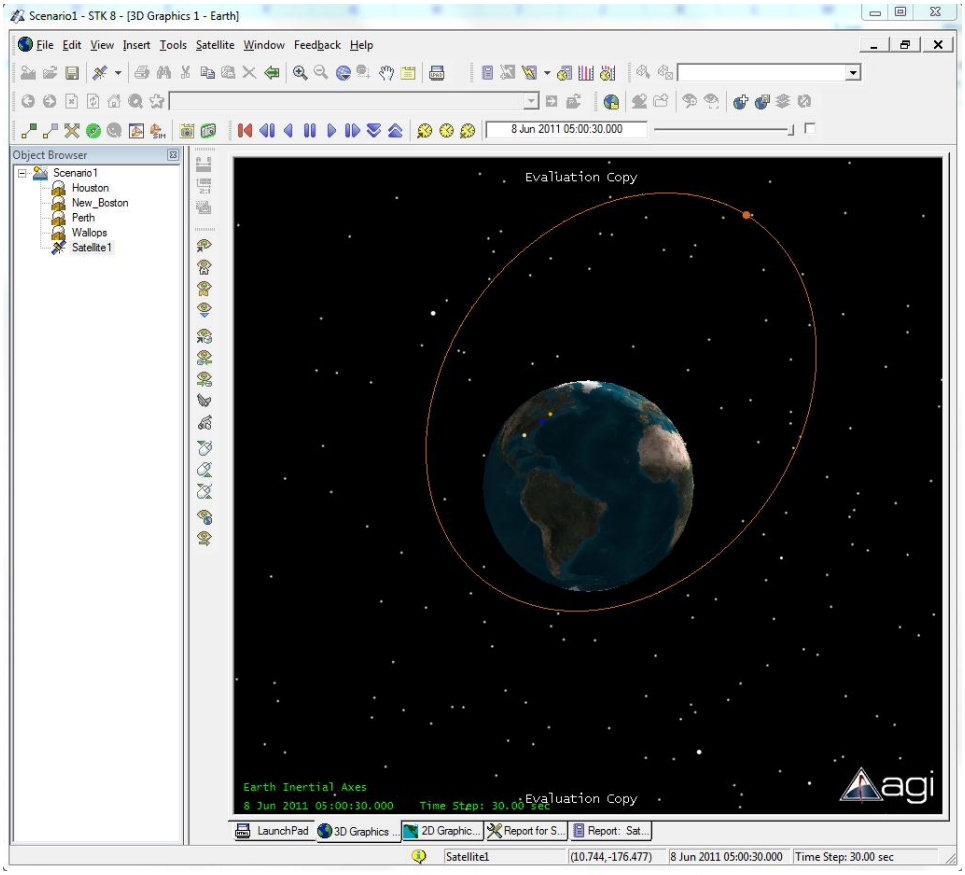

Figure 3-3. STK interface.

Within the perspective of Bouquet and FC development, STK offers all visualization, simulation, editing and optimization capabilities related to satellites. It

also offers the capability to run numerous satellites and especially group them into constellations, after which the complex interactions between the objects can be studied. However, STK is yet to implement the FC theory specifically, and thus the capability to elegantly construct and optimize FCs using the orbital parameters or lattice framework is missing.

With STK having proprietary source code, it's impossible to investigate whether it has the necessary internal architecture to allow optimization and editing of FCs. However, it shouldn't be declared that STK will never implement FC theory in the future. If FC theory gains enough usage it's likely that the developers of STK will add a comprehensive FC analysis tool as well, directly overlapping with the domain of Bouquet.

Table 3-3. Comparison of STK and Bouquet capabilities.

Capabilities (Detailed)	**STK**	**Bouquet**
3D visualization	Yes	Yes
2D projection visualization	Yes	Yes
Draw trajectories	Yes	Yes
Satellite constellation analysis	Yes	Yes
FC specific construction, editing, and optimization	No	Yes
Planet interchangeability	Yes	Yes
Draw coordinate frames	Yes	Yes
Arbitrary relative coordinate frames	Yes	Yes
Arbitrary relative trajectories	Yes	Yes
Incompatible orbits	Yes	Yes
Sun and day/night simulation	Yes	Yes
Planet axial rotation simulation	Yes	Yes
Basic time controls (e.g., stop, play)	Yes	Yes
Advanced time controls (e.g., skip, initial time, final time)	Yes	Yes
Preferences controls (e.g., adjust colors, visibility, units)	Yes	Yes
Texture rendering	Yes	Yes
Other advanced STK capabilities [1]	Yes	No

[1] The many advanced STK capabilities that Bouquet, in its current version, doesn't possess are grouped into one category for brevity. Examples include data reporting, attitude modeling and sensor modeling.

Table 3-3 illustrates that the focus of Bouquet, in its current version, on FCs is its only technically defining feature when compared to STK. The distinction can be eliminated if AGI decides to implement FC theory into STK which should be assumed to happen as discussed above. However, this doesn't mean that Bouquet has to become obsolete. One of the goals of Bouquet is to provide a free open source alternative that is developed collaboratively by many engineers, scientists and programmers around the world, leading to continuous improvement over time and a competitive product.

3.4 GMAT: General Mission Analysis Tool

The General Mission Analysis Tool (GMAT) is an open source space mission analysis tool designed by NASA in collaboration with private industry. Although it doesn't currently offer specific LFC analysis and simulation, it has extensive tools to simulate satellite orbits, design and optimize missions and customize using scripts [6]. One of its strengths is its ability to read MATLAB m-files and script routines using m-files. Figure 3-4 illustrates the GMAT interface.

Its capabilities compared to Bouquet are similar to the STK comparison in Section 3.3. However, GMAT lacks the 2D ground track projection display and its user interface is heavily dependent on scripting methods, thus decreasing ease of use. Due to its open source nature, the goal of combining the functionality of Bouquet and GMAT in the future is an interesting possibility.

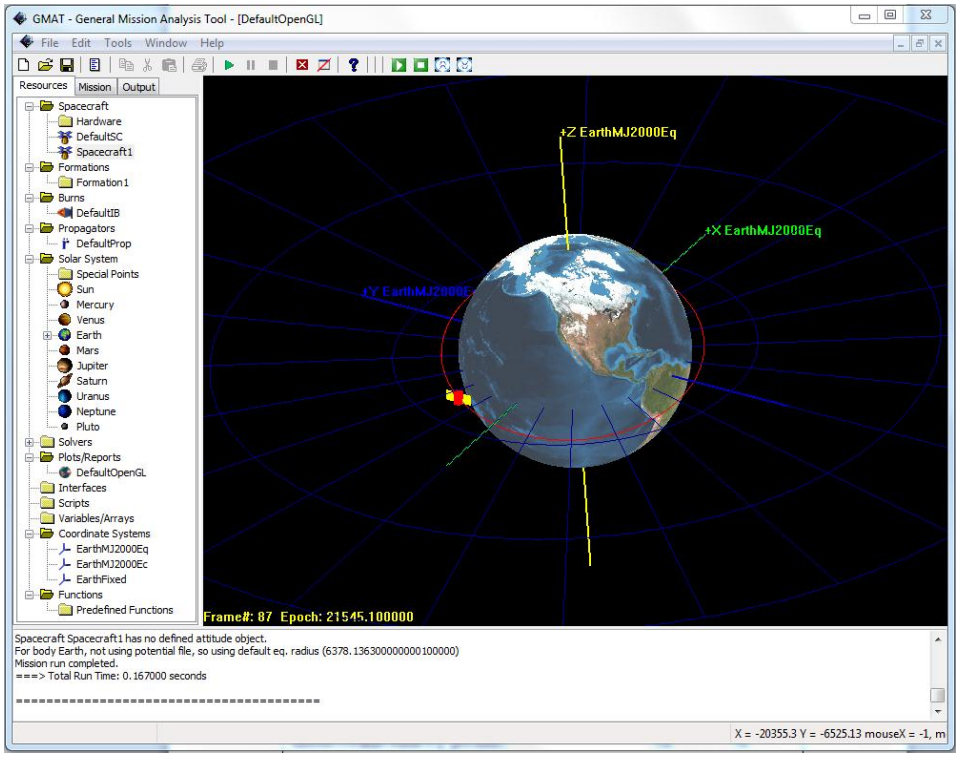

Figure 3-4. GMAT interface.

3.5 Google Earth: Virtual Globe Type Software

Virtual globe software, most famously exemplified by Google Earth (Figure 3-5), are applications with a primary focus on navigation and exploration. They all provide a virtual planet, usually Earth, displayed in 3D and allow the user to interact with it in a multitude of ways.

Compared to Bouquet, virtual globe software are highly focused on the planet and its systems. Capability such as detailed street maps, satellite surface imagery, display of borders and landmarks, and 3D terrain and buildings are included [7]. This kind of focus is inherently not the goal of Bouquet and thus there is minimal overlap in capability. However, many individually relevant ideas can be implemented into Bouquet

from virtual globe software. These ideas are discussed below.

The analysis focuses on Google Earth specifically as it is the most widely used and well developed example. Table 3-4 lists out the capability areas of Bouquet and Google Earth to illustrate the non-overlapping nature of each application.

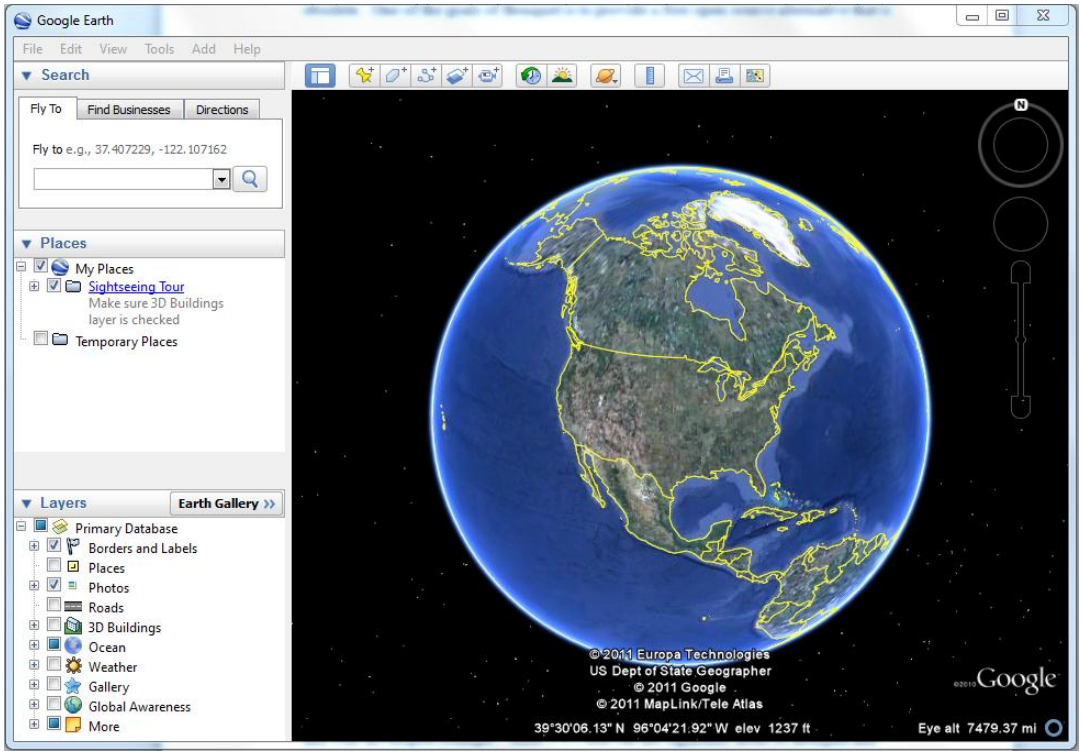

Figure 3-5. Google Earth interface.

The capabilities listed in italics are ideas from Google Earth that can be extremely useful to Bouquet and any major software development in general. Collaborative knowledge development encompasses Google Earth's ability to have its users tag landmarks and businesses, design 3D buildings and terrain, and overlay complex information such as demographics, traffic, and natural resources to name a few.

Any software becomes a powerful tool when its users can contribute their collective knowledge into its improvement. Google Earth also enables anyone to develop plug-ins for it using JavaScript, further giving users the opportunity to expand it towards their needs.

Table 3-4. Comparison of Google Earth and Bouquet capabilities.

Capability (Broad)	**Bouquet**	**Google Earth**
Detailed street maps and images	No	Yes
Detailed satellite surface images	No	Yes
Terrain features	No	Yes
Buildings, landmarks and businesses	No	Yes
Borders, roads and infrastructure	No	Yes
Tools: navigator, ruler, paths, image overlay	No	Yes
Sun positioning and day/night distinction	Yes	Yes
Simulation over any time frame	Yes	Yes[1]
Satellite simulation	Yes	No[2]
Constellation simulation	Yes	No
Inherent coordinate frames (e.g., geocentric)	Yes	No
User defined coordinate frames	Yes	No
Display 2D projection	Yes	No
Planet interchangeability	Yes	No[3]
Collaborative knowledge development	Not yet	Yes
Scripting language support	Not yet	Yes

[1] For the Sun, Moon, Mars and Earth interaction only. Limited to visualization.

[2] Only predefined major satellites or spacecraft (e.g., ISS, shuttle, Hubble).

[3] Limited to the Sun, Moon and Mars. No custom bodies.

Many other virtual globe software exist besides Google Earth (most of them open

source). Marble, NASA World Wind, Bing Maps, SkylineGlobe and Microsoft Virtual Earth 3D comprise a brief non-inclusive list.

3.6 Celestia: Planetarium Type Software

As with virtual globe software, planetarium type software presents another category of applications that don't overlap with Bouquet's focus. Planetarium software, represented here with the Celestia program, focus on visualizing and simulating the greater cosmos. Usually having access to huge databases of celestial objects, these programs allow the user to freely fly to and observe countless objects ranging from a tiny asteroid to clusters of galaxies [8].

Unlike Bouquet, planetarium software doesn't offer detailed simulation of satellites apart from predefined satellite paths such as the Voyagers or ISS. Table 3-5 illustrates the distinct capabilities of the two programs.

Capabilities in italics indicate ideas that may useful in the future development of Bouquet. Having a large database of stars could be used to simulate star tracking instruments and attitude control. The collaborative knowledge and scripting language support provide the same benefits as discussed in Section 3.4. Celestia offers users the ability to add new objects and features using the Lua language.

The list of other known planetarium software include Google Sky, Digital Universe Atlas, KStars, Stellarium, Cartes du Ciel, Worldwide Telescope, Xephem, Starry Night, Redshift, Thesky and Universe Sandbox.

Table 3-5. Comparison of Celestia and Bouquet capabilities.

Capability (Broad)	**Bouquet**	**Celestia**
Large database of celestial objects	Not yet	Yes
Visualize solar systems, galaxies and clusters	No	Yes
Visualize asteroids, moons, comets and other celestial objects	No	Yes
Free camera flight	No	Yes
Sun positioning and day/night distinction	Yes	Yes
Simulation over any time frame	Yes	Yes
Satellite simulation	Yes	No[1]
Satellite constellation simulation	Yes	No
Inherent coordinate frames (e.g., geocentric)	Yes	No
User defined coordinate frames	Yes	No
Display 2D projection	Yes	No
Planet interchangeability	Yes	Yes
Collaborative knowledge development	Not yet	Yes
Scripting language support	Not yet	Yes

[1] Only predefined major satellites or vehicles (e.g., ISS, shuttle, Hubble).

4. BOUQUET: APPLICATION OVERVIEW

4.1 Initial Requirements

The initially planned specs of Bouquet are listed below.

System:

- OS: Windows 7, Vista and XP.
- Graphics: any 3D display capability.
- Hard disk memory usage: undefined.
- RAM usage: undefined.

Displays:

- 3D orbital display: coordinate frames, satellites, planet, sun, shadows, relative trajectories (multiple), inertial trajectories and surface texture.
- 2D projected display: Mercator projection.
- Processing load: smoothly simulated animation (30+ fps) for at least 100 simultaneous satellites.
- Display controls: zoom, pan and rotate.

Timing:

- Precision: precision up to milliseconds.
- Units: JD and dates.

- Functionality: speed up, slow down, pause and stop.

Orbital Mechanics:

- Satellite orbit calculation architecture: handful of predefined orbit models.
- Predefined orbit models available: Keplerian, sgp4, sgp8, sdp4 and sdp8.
- Perturbation handling: J_2.
- Planet: simulate axial rotation and sidereal angle according to given date.
- Sun: simulate position according to date.

Flexibility:

- Custom objects: satellites and constellations.
- Custom and interchangeable models: interchangeability between predefined satellite orbit models.
- Flexible construction methods: constellations.

Environment:

- Color adjustment: background coordinate frames and trajectories.
- Units: metric and imperial.
- Sizing: undefined.
- Display and information areas: 3D display, 2D display, object selection pane and properties pane.

- Display area adjustment: ability to hide or show selection and properties panes.
- Session continuity: ability to save and load sessions.

4.2 Methodology

4.2.1 Selection of Development Environment, Language and Graphics API

The important process of selecting the software development kit (SDK), programming language and graphics API was based on several factors including the author's personal skill-set, experience and cross-platform development considerations. Final decisions set Qt (along with its built-in GUI developer) as the SDK, C++ as the programming language and OpenGL as the graphics API.

At the outset, OpenGL was decided to be used as the graphics API due to its open source nature, cross-platform functionality, ease of development and wide use in similar applications such as Google Earth and Celestia [7-9]. Table 4-1 compares OpenGL with DirectX. This decision in turn meant Bouquet was best programmed in C++ as it was the primary language to utilize OpenGL (other languages are possible, but technically troublesome and time-consuming due to another layer of “translation” structure being necessary). The slower pace of updates for OpenGL in industry was not considered an issue as Bouquet does not aim to be on the cutting edge of graphics technology.

Table 4-1. Comparison of OpenGL and DirectX graphics APIs.

	OpenGL	**DirectX**
Pros	Open source. Cross-platform. Widely used in similar programs. Easier learning and development [9].	Easier integration with Microsoft Visual Studio and Windows Presentation Foundation. Updates reflected in industry quickly.
Cons	Practically limited to C++. Updates reflected in industry at slower pace.	Proprietary. Windows platform only. Harder learning and development.

Initial SDK consideration went to Microsoft Visual Studio (VS) [10], with its Windows Presentation Foundation (WPF) used for GUI development [11], to program Bouquet, largely due to the author's lack of experience with other environments. However, there were a number of problems with this choice. First, WPF is primarily designed to be used with C# as the language, conflicting with the use of C++ for OpenGL. Second, use of WPF necessitates use of Microsoft's .NET framework, limiting the development of a cross-platform product (.NET framework support incomplete on other platforms). Third, not using WPF means reverting to the older Windows Forms GUI development method, which is based on old standards and heavily deals with Microsoft's troublesome MFC and MSDN libraries. Fourth, integrating OpenGL with WPF is technically tedious and challenging [12].

Fortunately, Dr. Christian Bruccoleri suggested Qt, an open source SDK supported by a Nokia. Qt is not only open source, but also cross-platform, C++ based, seamlessly integrates with OpenGL and has excellent GUI development classes that save the programmer from dealing with Windows' internal MSDN classes [13, 14]. Table 4-2 compares Qt and Microsoft Visual Studio.

Table 4-2. Comparison of Qt and Microsoft Visual Studio.

	Qt with built-in GUI developer	**Microsoft VS with WPF**
Pros	Open source. Cross-platform. Easier learning and development. Wide range of convenience classes. Excellent GUI development structure. Seamless integration with OpenGL. Extensive documentation available.	Easier integration with DirectX. Well known and widely used. Extensive documentation available.
Cons	Practically limited to C++. Harder integration with DirectX.	Proprietary. .NET framework effectively Windows only. Harder integration with OpenGL.

Language selection was primarily dictated by other considerations as discussed above. C++ is a lower level language than C#, and hence considered more "powerful" due to the greater control given to programmers [15]. For the purposes of Bouquet, however, both languages provide the necessary Object Oriented Programming (OOP) capabilities. Other languages such as Java were not considered due to lack of experience in those languages.

4.2.2 Application Internal Architecture

The development of Bouquet uses extensive OOP principles, such that the modularity of objects and classes maximizes flexibility in future development. Each class is attempted to closely resemble an intuitively logical framework, while weighing the effects of performance and coding. Several classes make up the core framework of Bouquet as illustrated in Table 4-3.

Using the mechanism of pointers in C++, most objects simply "observe" the other objects and their states and change themselves accordingly whenever any of the referenced states change. This type of object interaction structure avoids resource

intensive copying and manipulation of data and maintains a flexible modular structure.

Table 4-3. Bouquet Core Classes.

Class Name	Description
SatelliteClass	Stores all orbital info for given satellite, most important being initial and current positions and velocities. Contains the orbital trajectory classes as its children.
ConstellationClass	Based on the IDSatContainer class, this class stores a number of satellites as a constellation group and allows ID based organization, i.e., free insertion, removal, and storing of metadata. It also handles FC specific operations such as proper display of relative trajectories.
SunClass	Stores Sun data and calculates its orbital movement given a specific planet.
PlanetClass	Stores predefined planet data and describes a given planet's orbital characteristics and display methods.
MainWindow	Core class of Bouquet, launches first and organizes all other classes and GUI functionality as its children.
FCTimer	Maintains timing functionality such as keeping track of initial time, current time and final time.
OrbitView	The only class that implements OpenGL code, it is responsible for handling all 3D display capability. Can be instantiated multiple times for multiple 3D views.
MapScene	Responsible for creating and maintaining a 2D scene. Together with MapView, provides the 2D projected display functionality of Bouquet.
MapView	Offers multiple views onto a 2D scene. Together with MapScene, provides the 2D projected display functionality of Bouquet.
TransformMatrix	Handles the functionality of arbitrary rotating coordinate frames.
InertialFrame	Handles the functionality of predefined inertial coordinate frames, i.e., geocentric frame.
TrajectoryMatrix	Given a specific satellite and inertial frame, it stores the inertial orbit trajectory data.
RelativeMatrix	Given a specific satellite and an arbitrary rotating coordinate frame, it stores the data of the trajectory as seen from said coordinate frame.
ModelPrototype	Abstract class that allows different orbit propagation methods to be defined through inheritance.
ProjectionPrototype	Abstract class that allows different 3D to 2D projection methods to be defined through inheritance.

4.2.3 Matrix Manipulation Tools

The primary internal tools created for Bouquet are the matrix manipulation classes WMatrix and WVector, both of which were developed in-house as opposed to

outside sources. WMatrix handles all matrix manipulation and calculation jobs requested by Bouquet, while WVector is an inaccessible class used internally by WMatrix.

Table 4-4. WMatrix syntax and MATLAB syntax.

Operation	**WMatrix**	**MATLAB**
Addition	C = A + B; C = a + B;	C = A + B; C = a + B;
Subtraction	C = A – B; C = a – B;	C = A – B; C = a – B;
Matrix multiplication	C = A*B; C = a*B;	C = A*B; C = a*B;
Unary minus	C = -A;	C = -A;
Transpose	C = A.transpose();	C = A'; Also handles complex conjugate.
Vector to skew-symmetric	C = A.skew(); A must be 3x1 or 1x3.	Undefined.
Dot product	c = A.dot(B);	c = dot(A, B);
Cross product	C = A.cross(B); A and B must be 3x3.	C = cross(A, B); A and B must be 3x3.
Scalar element selection	c = A(); if A is 1x1. c = A[i]; if A is a vector. c = A(i, j); if A is any dim.	c = A; if A is 1x1. c = A(i); if A is a vector. c = A(i, j); if A is any dim.
Group selection	C = A(r1, r2, c1, c2);	C = A(r1 : r2, c1 : c2); C = A(r1 : r2, :); C = A(:, c1 : c2); C = A(r1 : r2); if A is a vector.
Scalar element overwrite	C() = a; if C is 1x1. C[i] = a; if C is a vector. C(i, j) = a; if C is any dim.	C = a; if C is 1x1. C(i) = a; if C is a vector. C(i, j) = a; if C is any dim.
Group overwrite	C = A; if A and C same size. C(i, j, A); i and j define element in C corresponding to A(0, 0).	C = A; if A and C same size. C(r1 : r2, c1 : c2) = A; C(r1 : r2, :) = A; C(:, c1 : c2) = A;
Concatenate	C = A.resize(ra, ca + cb)(0, ca, B); horizontally. C = A.resize(ra + rb, ca)(ra, 0, B); vertically. ra is number of rows in A. ca is number of columns in A. rb is number of rows in B. cb is number of columns in B.	C = [A B]; horizontally. C = [A; B]; vertically.
Element-by-element power	C = A.pow(b);	C = A.^b;

With the author's user experience of MATLAB in mind, the two classes were developed to give similar functionality to applications written in C++ (in contrast to m-files). Performance considerations, e.g., reducing the number of expensive actions such as copying large fields of data, were implemented as much as possible.

The key strengths of the WMatrix class are in easily accessing and setting elements and manipulating their positioning. The class uses extensive operator based functions and overloaded functions to interpret the user's code in the most intuitive manner possible. Table 4-4 shows some of the WMatrix syntax in comparison to MATLAB syntax.

On top of the benefit of minimal syntax necessary to manipulate matrices, the functional similarity to MATLAB is advantageous in converting m-files to C++ files. The WMatrix class also has built-in assert checks to confirm that the size of matrix inputs are correct and prevent uncaught overflow. Matrices can be constructed using either an element-by-element approach, imported from a QVector object, or using a pointer to an array. Utility functions such as extrapolation, binary search, insert and delete are also offered within the class.

Performance considerations involved in WMatrix and WVector development are:

- Reference passing when possible.
- Keeping track of matrix states (size and type) to reduce unnecessary operations.
- Rows constructed of sub-vector objects to minimize copy operations.

The matrix manipulation tool is not only valuable to Bouquet, but can also be distributed individually for use in other C++ programs. Standalone improvement and support can also be expected.

4.2.4 3D Display Architecture

The OrbitView class handles all 3D display responsibilities within Bouquet and is the only class with OpenGL code. The class inherits the QGLWidget class, provided by Qt, so that it can be embedded into the display widget within the application window. Appropriate virtual functions are reimplemented for the drawing capabilities. Multiple OrbitView objects can be instantiated at the same time to offer different views or visualize different scenarios altogether.

Most importantly, OrbitView is programmed such that it can display any planet, satellite, frame and trajectory configuration without any adjustment to itself. In other words, only the planet, satellite, frame and trajectory classes need to change states and the observing OrbitView objects automatically reflect the changes (Figure 4-1). With this setup, the main visualization coding is decoupled from other processes, affording more flexibility and ease of editing with its use. Individual OrbitView objects can then be created and moved around to other parts of the program as necessary. An example of this use is given in Figure 4-2, where the 3D preview display in the satellite creation dialog uses another instance of the OrbitView class with no need to rewrite its code.

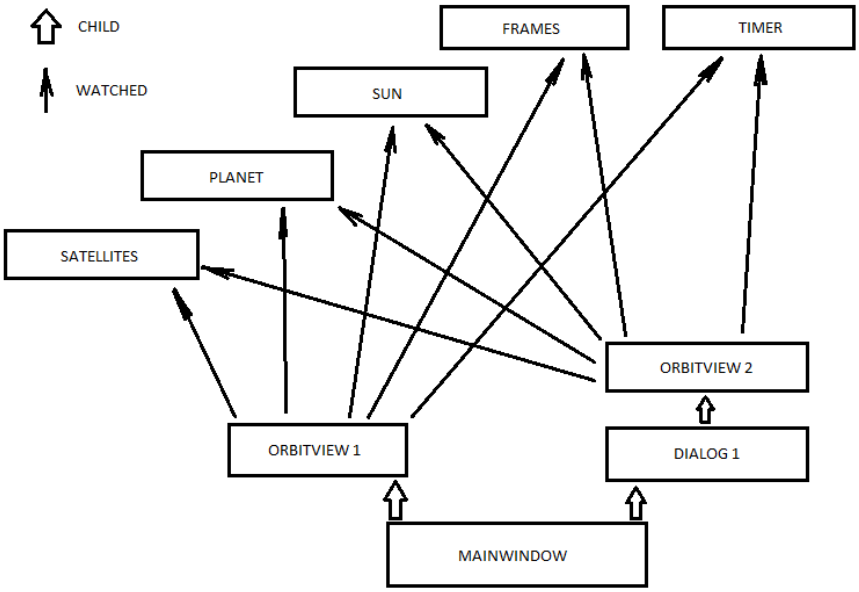

Figure 4-1. Simplified OrbitView dependency diagram.

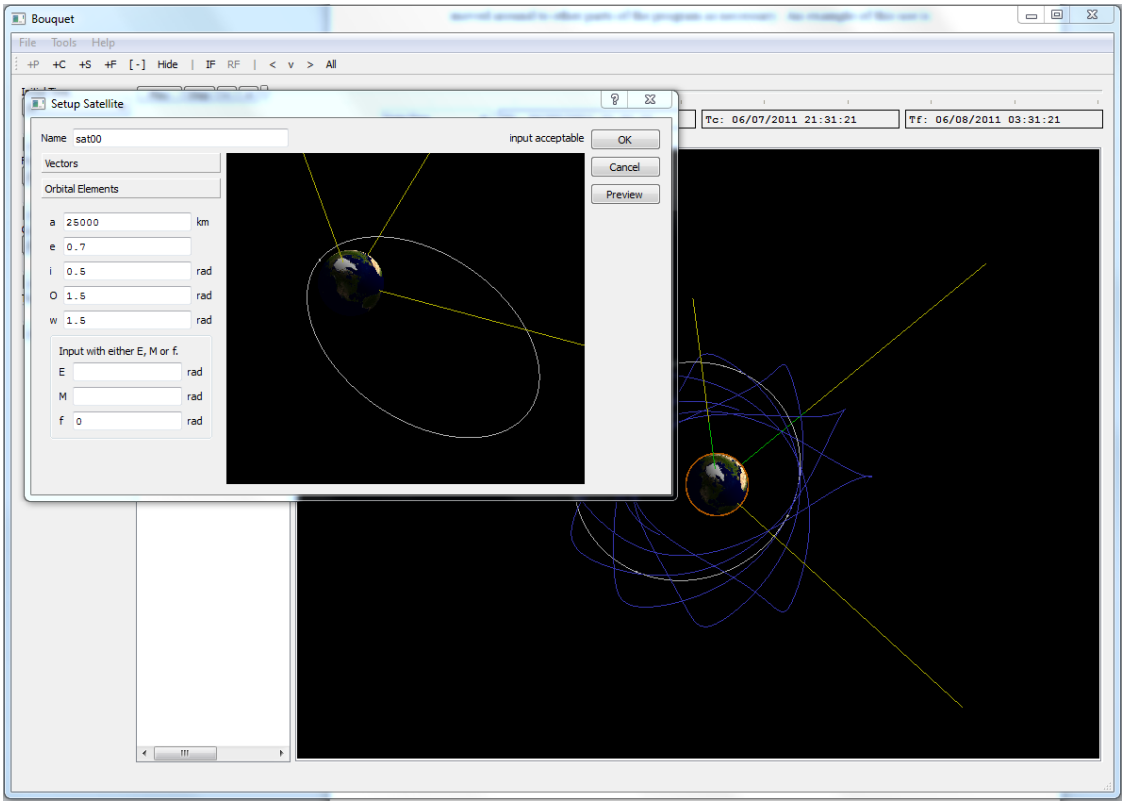

Figure 4-2. Satellite creation dialog.

4.2.5 2D Projection Display Architecture

The display of the 2D projections of orbits has two phases in its construction. First, the internal simulation model is converted from 3D to 2D using a given projection method. Second, the converted 2D data points are displayed using specialized classes in Qt.

The conversion from 3D data points to 2D data points utilizes an arbitrary projection function such as equirectangular or Mercator. Using polymorphic programming concepts, different projection methods can be implemented by inheriting from the abstract class ProjectionPrototype and reimplementing the *getProjections()* function (Figure 4-3). The addition of a correspondingly projected planet surface texture file is also necessary.

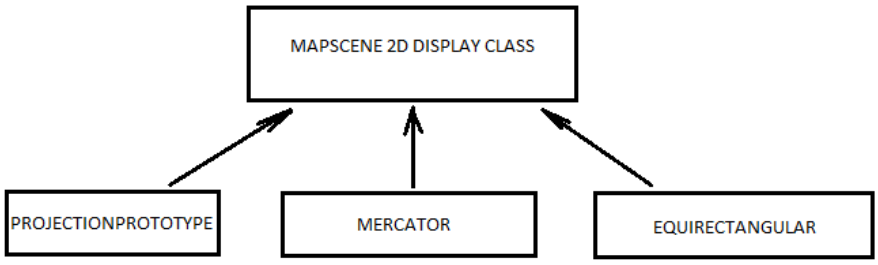

Figure 4-3. Diagram of polymorphism in action. The MapScene class is only designed to handle ProjectionPrototype classes, but since the Mercator projection and equirectangular projection are both derived from ProjectionPrototype, they too can be passed onto MapScene.

The 2D display architecture of Bouquet is based on an entirely different method than the 3D method. In contrast to the 3D display, the 2D display uses the scene, item, and view methodology provided in Qt, without any use of OpenGL (Figure 4-4). This

methodology is only capable of 2D views. There are several advantages and disadvantages associated with this choice.

Advantages:

- Already written, tested and stable classes offered by Qt.
- Separation of the scene, items and view responsibilities offering flexibility and modularity.
- Ability to handle large numbers of items and multiple views.

Disadvantages:

- Difficult animation framework using multiple sliding layers. Especially troublesome with non-cylindrical projections.
- Display responsibilities relegated to individual items involve a convoluted process: creating, tracking, modifying and destroying items properly.

Despite the advantages offered, in hindsight it may have been better to develop a custom class that more smoothly animates the projected data. Most importantly, the ability to use non-cylindrical projections will require an alternative method of display.

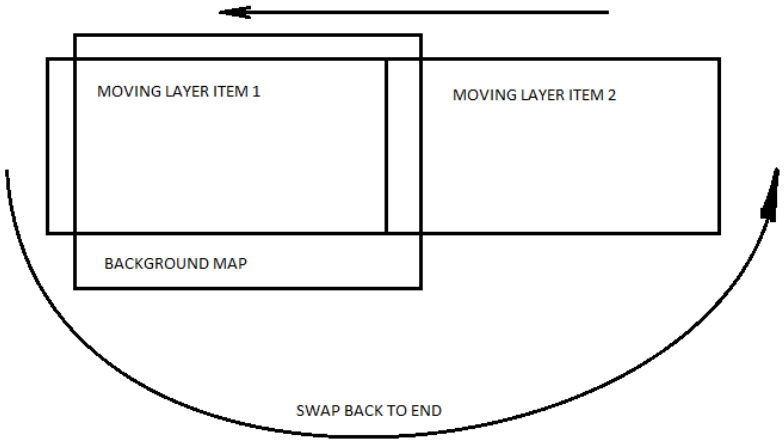

Figure 4-4. Diagram of 2D layer mechanism. Each layer acts like a rotating frame with respect to the planet and items on those layers don't have to consider any of the rotation calculations. Layer 1 and Layer 2 are identical except for the offset.

4.3 Capabilities

4.3.1 Fulfillment of Objectives

This section seeks to demonstrate the current capabilities of Bouquet and compare them against the initially set out requirements to get an overview of whether objectives were successfully completed. While any computer application can always be further improved, Bouquet is no exception, Table 4-5 illustrates that most of the initial objectives for the program have been achieved.

The current capabilities of Bouquet can best be explained in the visualization, simulation, editing and optimization framework introduced in Section 1.

Table 4-5. Comparison of initial requirements and actual capabilities.

Specs	Initial Requirements	Actual Capabilities	Met?
System			
OS	Windows 7, Vista and XP.	Windows 7, Vista, XP and possibility of additional OSs in the future.	Yes
Graphics	Any 3D display capability.	Hardware accelerated OpenGL.	Yes
Hard disk memory usage	Undefined.	<>get rom usage>	Yes
RAM usage	Undefined.	<>get ram usage>	Yes
Displays			
3D orbital display	Coordinate frames, satellites, planet, sun, shadows, relative trajectories (multiple), inertial trajectories and surface texture.	Coordinate frames, satellites, planet, sun, shadows, relative trajectories (multiple), inertial trajectories and surface texture.	Yes
2D projected display	Mercator projection.	Mercator projection, rectilinear projection and projection type interchangeability.	Yes
Processing load	Smoothly simulated animation (30+ fps) for at least 100 simultaneous satellites.	At least 30 fps for 100 simultaneous satellites.	Yes
Display controls	Zoom, pan and rotate.	Zoom, pan and rotate.	Yes
Timing			
Precision	Precision up to milliseconds.	Precision up to milliseconds.	Yes
Units	JD and dates.	JD, dates, seconds, months, days and years.	Yes
Functionality	Speed up, slow down, pause and stop.	Speed up, slow down, pause, stop and increment.	Yes
Orbital Mechanics			
Satellite orbit calculation architecture	Handful of predefined orbit models.	Switchable orbit models and ability to add custom models.	Yes
Predefined orbit models available	Keplerian, sgp4, sgp8, sdp4 and sdp8.	Keplerian and fast Keplerian.	No
Perturbation handling	J_2.	Not yet implemented.	No
Planet	Simulate axial rotation and sidereal angle according to given date.	Simulate axial rotation and sidereal angle according to given date.	Yes
Sun	Simulate position according to date.	Simulate position according to date.	Yes
Flexibility			
Custom objects	Satellites and constellations.	Planet, coordinate frames, satellites and constellations.	Yes

Table 4-5 continued.

Specs	Initial Requirements	Actual Capabilities	Met?
Custom and interchangeable models	Interchangeability between predefined satellite orbit models.	Satellite orbit models and 2D projection models.	Yes
Flexible construction methods	Constellations.	Satellites, constellations and coordinate frames.	Yes
Environment			
Color adjustment	Background coordinate frames and trajectories.	Background, highlights, trajectories, satellites, and coordinate frames.	Yes
Units	Metric and imperial.	Metric and imperial.	Yes
Sizing	Undefined.	Adjustable main window, display fields, information panes and dialog boxes.	Yes
Display and info areas	3D display, 2D display, object selection pane and properties pane.	Multiple 3D displays, multiple 2D displays, tree-based object selection pane, table-based properties pane, text-based utility and status console, and time manipulation pane.	Yes
Display area adjustment	Ability to hide or show selection and properties panes.	Multiple tabbed displays and ability to hide or show all panes.	Yes
Session continuity	Ability to save and load sessions.	Ability to save and load sessions.	Yes

4.3.2 Visualization

For 3D visualization, Bouquet is able to display a planet body along with an arbitrary number of orbiting satellites, each with their own inertial and relative trajectories (Figure 4-5). Multiple arbitrary coordinate frames are displayed and can be selected to be the stationary frame, with respect to the view, at any time. Day and night lighting on the planet is visible. Currently selected objects are highlighted and each object can be made hidden or visible as necessary. The background, colors of certain objects, highlighting details and line widths can be adjusted.

For 2D visualization, Bouquet is able to display arbitrary cylindrical projections,

provided the proper planet surface texture is available. The satellite body and its ground track are displayed (Figure 4-6). Highlighting, day and night lighting, and options adjustments are currently unavailable.

Zoom, rotate and pan controls are offered on both types of displays.

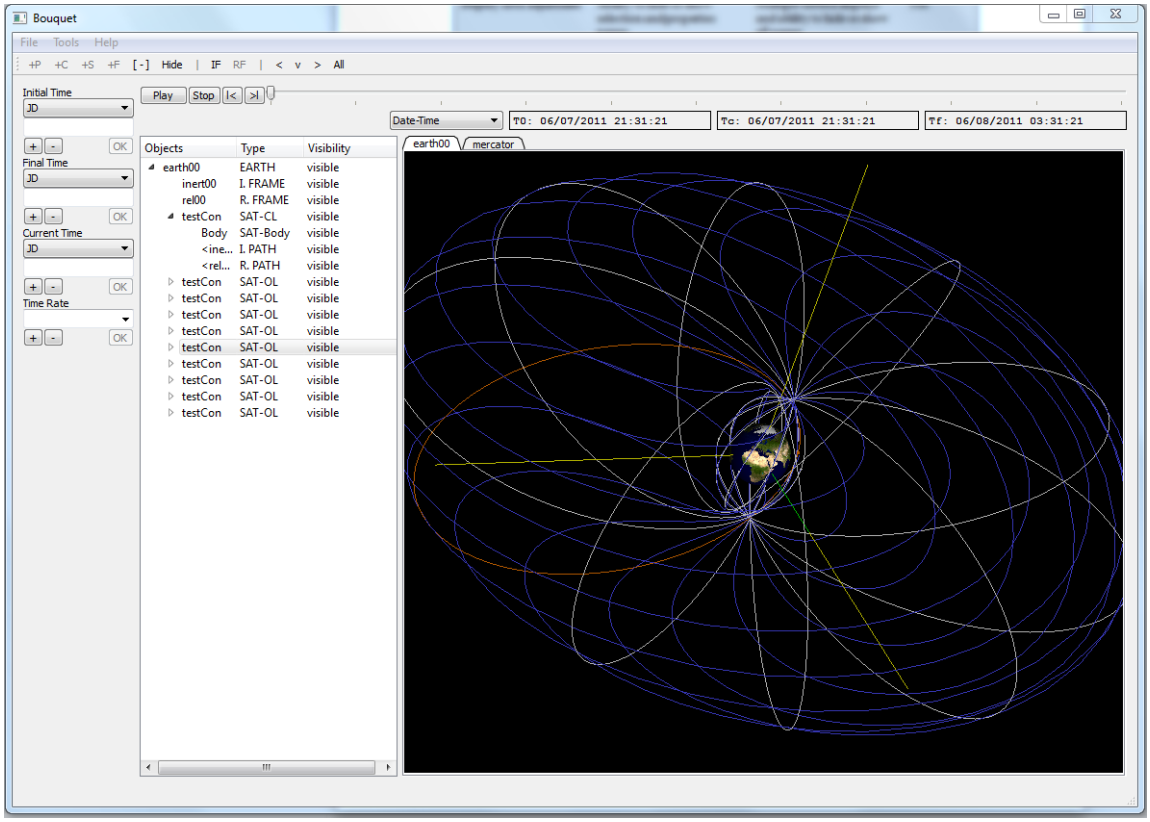

Figure 4-5. Bouquet 3D display.

4.3.3 Simulation

Bouquet has a central timer, encompassed in the FCTimer class, that keeps track of initial time, current time and final time of a given simulation. Initial time and final time are freely adjustable, using Julian date, calendar date, or local machine date

methods (Figure 4-7). Current time and final time can also be given in a delta time format from the initial time. After the times are set, Bouquet automatically adjusts the planet sidereal angle and sun position corresponding to the initial time. Initial states of all satellites and coordinate frames are considered to be at initial time.

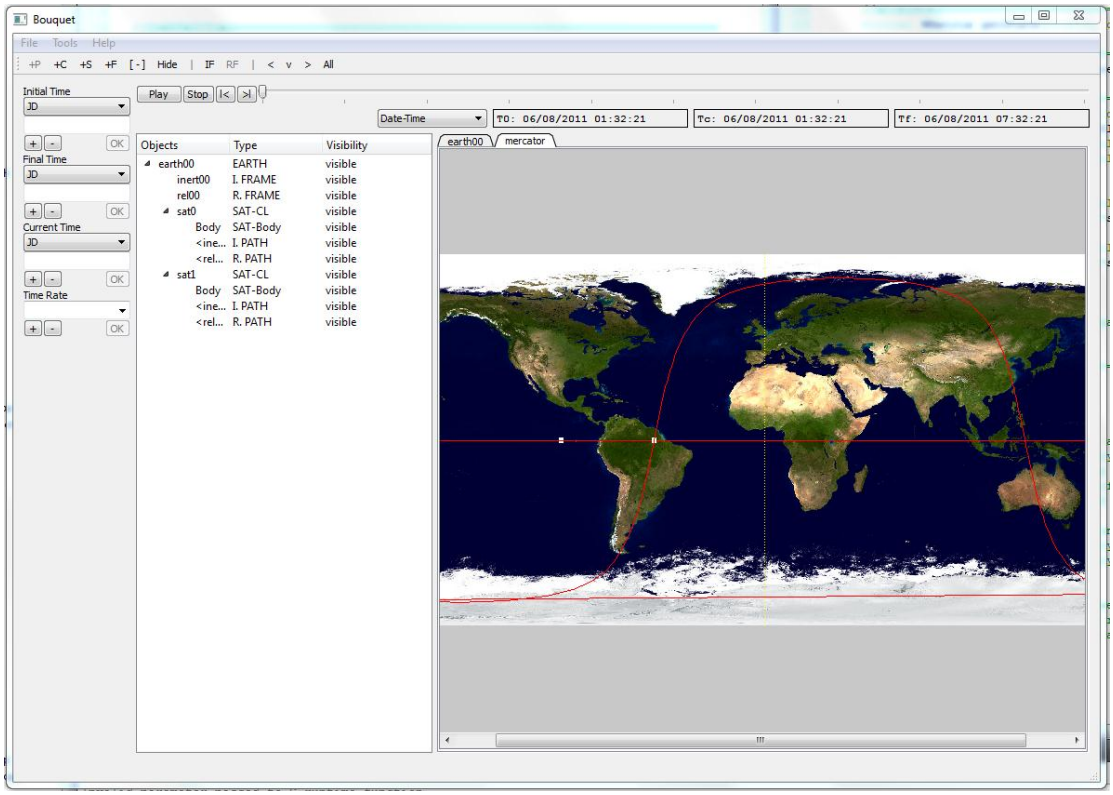

Figure 4-6. Bouquet 2D display.

Satellite inertial orbit trajectories are propagated once at the time of definition, using a set time interval between points. The array of points are stored for faster display and animation. Relative trajectories are obtained by transforming the inertial orbit to the corresponding relative frames. Compatible relative trajectories, which are closed, have

their point data stored for faster animation. Incompatible orbits have a continually changing data matrix, with new points calculated on the fly and appended at the front, while old points are deleted at the back. This setup minimizes calculations while retaining the ability to display incompatible orbits.

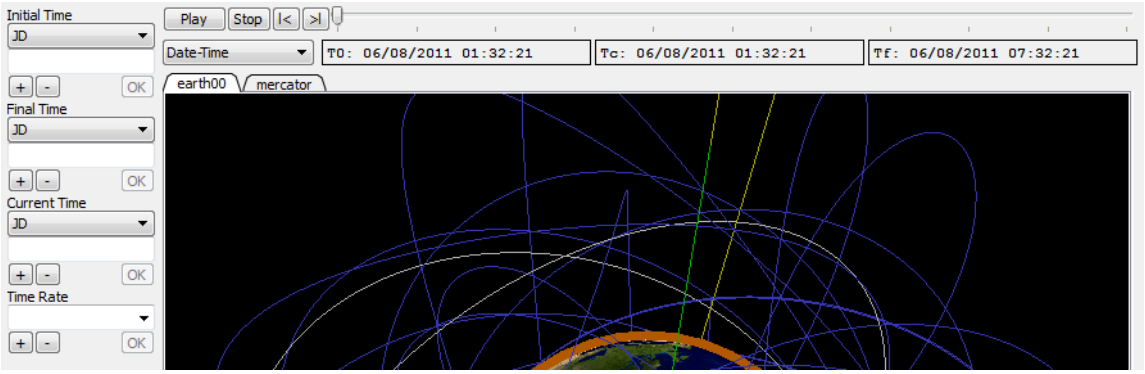

Figure 4-7. Bouquet time simulation controls in detail.

4.3.4 Editing

With flexibility an important goal, Bouquet minimizes hard-coded programming and offers many editing and custom input opportunities. The main planet body can be changed to predefined planets like Earth and Mars or set up with custom attributes.

Satellites can be defined with multiple methods, either using position and velocity vectors or orbital elements. The orbit propagation method can be edited by inheriting from the ModelPrototype abstract class and reimplementing its *getState()* function. The satellite class is built to easily allow future improvements such as attitude tracking and instrumentation considerations. FCs can also be defined using multiple methods and placeholders exist for future methods. Satellites can be deleted from and

added to FCs as necessary.

Completely arbitrary rotating coordinate frames can be defined using multiple input methods (Figure 4-8). The initial state of the frame can be defined by a transformation matrix or principle axis and angle, both with respect to the inertial frame. The rotation rate and axis can be defined to be about one of the orthogonal axis or with a custom vector.

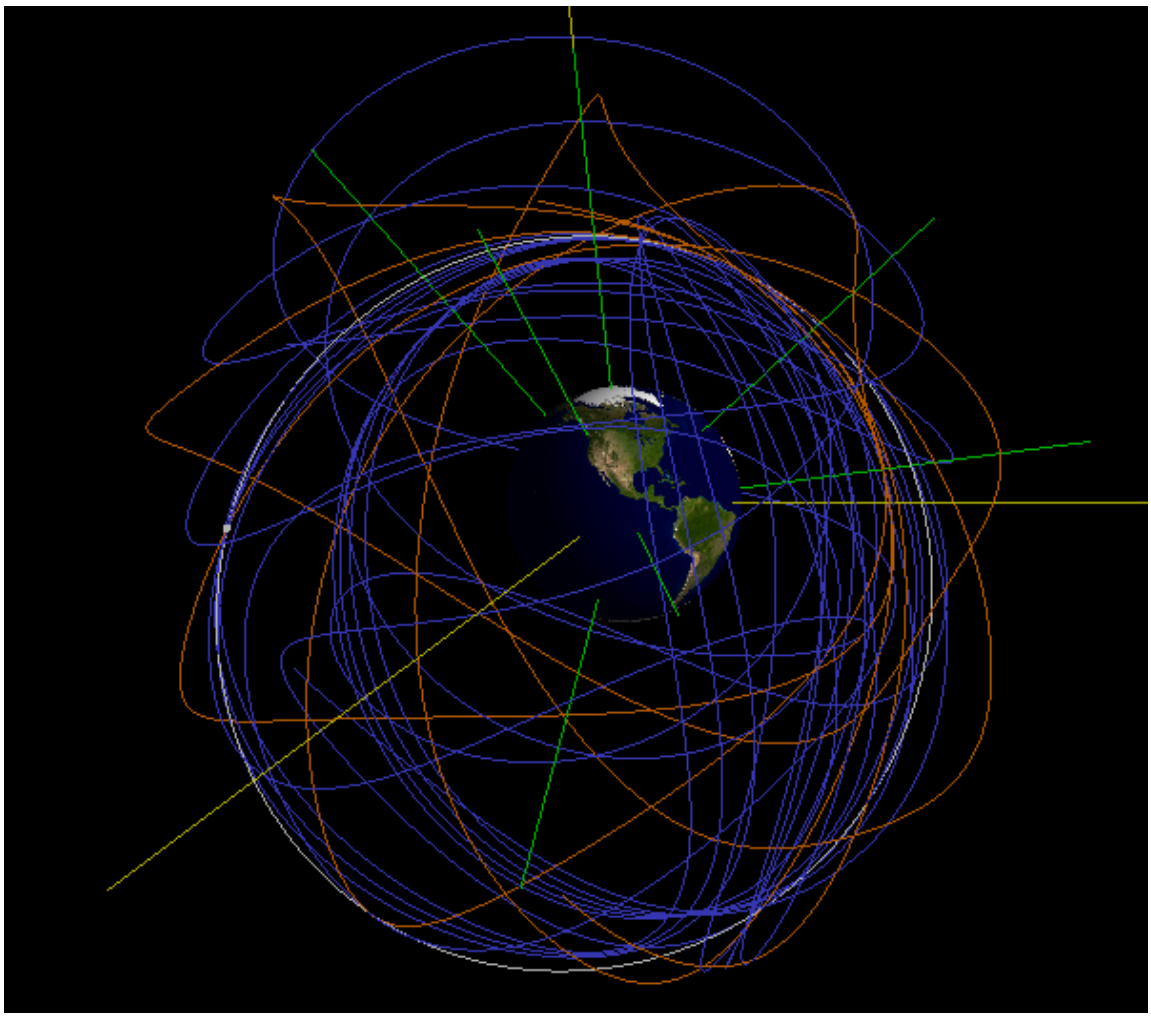

Figure 4-8. Multiple relative frames and trajectories. A single satellite with three relative trajectories displayed that each correspond to three different frames rotating about X, Y, and Z.

For the 2D display, the projection method can be edited as explained in Section 4.2.5. The framework is also programmed with future editing capability in mind, particularly with regards to instruments interacting between surface points and satellites.

4.3.5 Optimization

At this moment, optimization is the only pillar of operation that hasn't been achieved yet. Bouquet is designed with optimization of FCs in mind, so the basic framework necessary for optimization code is there. This framework includes the ability to add and delete satellites, FCs, and coordinate frames freely through automated code. But the actual optimization theories and code is not yet implemented. Research in this area of FCs is still ongoing and represents the most important utility of FCs and Bouquet in the future. Optimization is the link between tangible client needs and an appropriate FC fit for those needs.

5. TESTING AND VALIDATION

5.1 Code Testing

Continuous testing of the code was done during the development process to minimize major errors and, most importantly, to make debugging easier. Two different coding and testing methods were used over time: the piece by piece approach and the incremental expansion approach.

The piece by piece approach entails building one appropriately independent part of the program completely separately and then combines it with the main application afterwards. In other words, a new piece is coded, built and tested in its own standalone development environment and then connected to the actual application at the end. Figure 5-1 illustrates this method.

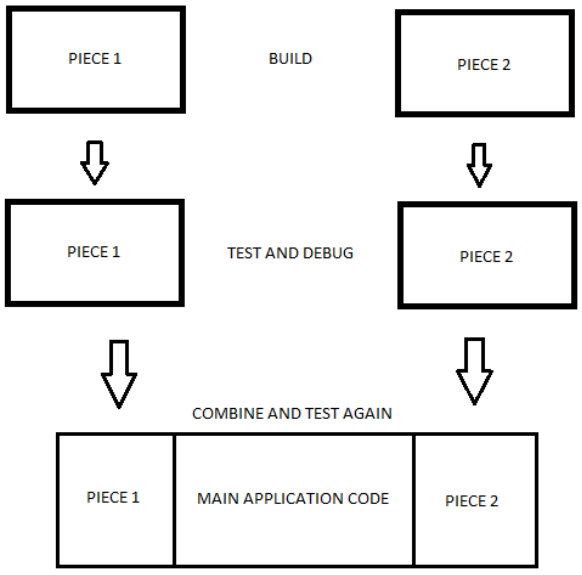

Figure 5-1. Piece by piece programming and testing approach.

In theory, this method modularizes the process such that bugs are confined to a small section and thus easier to correct. However, it was soon realized that the easier debugging aspect was marginal (combining pieces with the main code introduces errors itself and testing ability is limited with standalone pieces); instead the real benefit of this method being that many pieces can be developed simultaneously. As this actual benefit is useless with a single developer, the extra effort necessary to separate and combine pieces was not worth it and this development method was dropped for the incremental expansion approach.

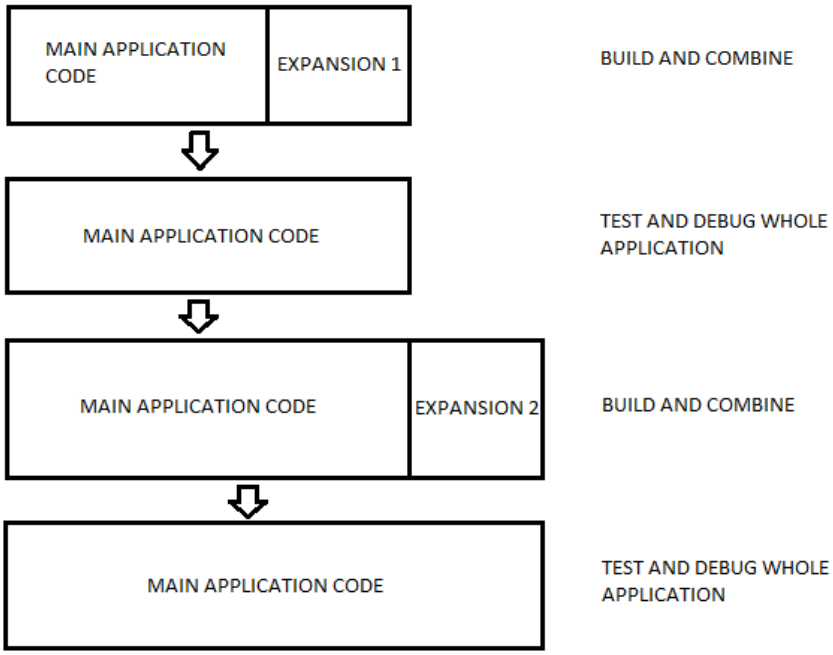

Figure 5-2. Incremental expansion approach.

As Figure 5-2 illustrates, the incremental expansion is a straight forward coding of the application in a single environment with all additions directly connected to the

existing code. The whole application is tested between each incremental addition. Although, as the application gets large, there is some risk of deeply embedded errors coming up that are difficult to solve, this approach is well suited for a single programmer and saves time in the long run, provided that assert statements are well used and unusual situations anticipated.

Testing of the application between expansions involves compiling and running it a handful of times, specifically focusing on triggering the most recently added code to run and display results, either through normal operation of the application or, if the code results in hidden effects, by forcefully displaying the results in a test window.

5.2 Platform Testing

Platform testing of Bouquet is limited to Windows XP, Vista and 7 at this time. Earlier versions of Windows are too rare and deprecated to worry about. Apple OS and Linux are worthy goals to consider in the future.

Bouquet works the best on Windows 7 since it is the platform that was coded in. The continuous testing and debugging during development ensures that problems on Windows 7 are minimized.

Testing on Windows XP is infrequent; about nine test runs have been done so far. There seem to be a high level of compatibility between Bouquet and Windows XP and unique problems – errors that don't occur on Windows 7 – haven't been found.

On the other hand, Windows Vista poses numerous problems for Bouquet, encountered in the span of roughly 20 test runs. Most of the errors are unique to Vista

and doesn't occur on 7 or XP, making debugging difficult as the author has limited access to Vista computers. Two prominent errors occur with Vista:

- Vista is missing or is unable to access a range of common image codecs used for texture files, including JPG, BMP, and TIFF, when they are loaded into an application. Thus Bouquet's globe texture doesn't draw. This problem can be fixed by bringing the codecs along with Bouquet or using PNG files, but for unknown reasons this fix is not guaranteed to work every time.
- Bouquet crashes when the Aero theme is disabled in Vista. Reasons are currently unknown.

5.3 Results Validation

To validate the results of Bouquet, preliminary testing was done by comparing its output to that of the industry proven STK. While small differences on the order of one percent or less existed, see Table 5-1, on the whole the results of Bouquet matched with STK and no wild outputs were present. Although not yet available, the ability to efficiently test Bouquet with large data sets will be possible after proper output formatting is implemented.

Table 5-1 and Figure 5-3 present a true anomaly comparison of a satellite orbit in STK and Bouquet, with all other fixed orbital elements being equal. Figure 5-3 indicates that error increases towards the end of the period and then resets to near zero when the period is completed. This effect could be due to cumulative round-off errors, which can

be improved with different propagation models and optimizing calculation methods. The satellite details used for this test can be found in Table A-1.

Table 5-1. Comparison of true anomaly between STK and Bouquet.

	True Anomaly (deg)			
Time	STK	Bouquet	Error	Error %
13:00:00.000	82.552	82.55175912	0.000240878	2.9179E-06
13:30:00.000	123.68	123.6987232	-0.01872318	-0.00015138
14:00:00.000	147.331	147.5068384	-0.175838441	-0.00119349
14:30:00.000	164.992	165.595689	-0.603688991	-0.0036589
15:00:00.000	180.563	182.0570705	-1.494070524	-0.00827451
15:30:00.000	196.207	199.4973329	-3.29033285	-0.0167697
16:00:00.000	214.137	215.8501213	-1.713121281	-0.00800012
16:30:00.000	238.514	232.5529869	5.961013076	0.0249923
17:00:00.000	281.957	278.4681058	3.488894247	0.01237385
17:30:00.000	16.236	16.21029383	0.025706173	0.00158328
18:00:00.000	94.271	94.3140097	-0.043009699	-0.00045623

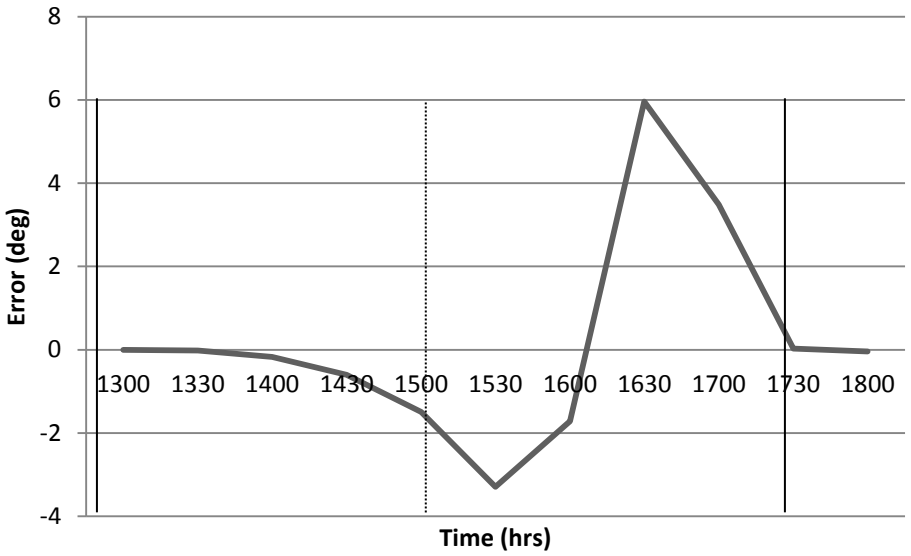

Figure 5-3. Difference of true anomaly between STK and Bouquet. Solid vertical lines indicate approximate perigee locations. Dotted vertical line indicates approximate apogee location.

Another satellite with a Molniya orbit was set up to compare the inertial position and velocity output to STK. The input details of this satellite along with the test output tables can be found in Tables A-2 to A-8. Figures 5-4 and 5-5 capture the percentage error comparison between the positions and velocities, respectively.

The position errors are not bad over all, with the sudden spike in the middle being an artifact of the illustration method caused by the position value becoming very small (in this case, the satellite was near x-y plane) in comparison to the error values, even though the error values themselves didn't increase much. The negative spike at the beginning of the velocity figure also results from this issue.

However, in the velocity error comparison figure, the exponential increase of the x velocity over the course of the test time frame (20 days) should be investigated.

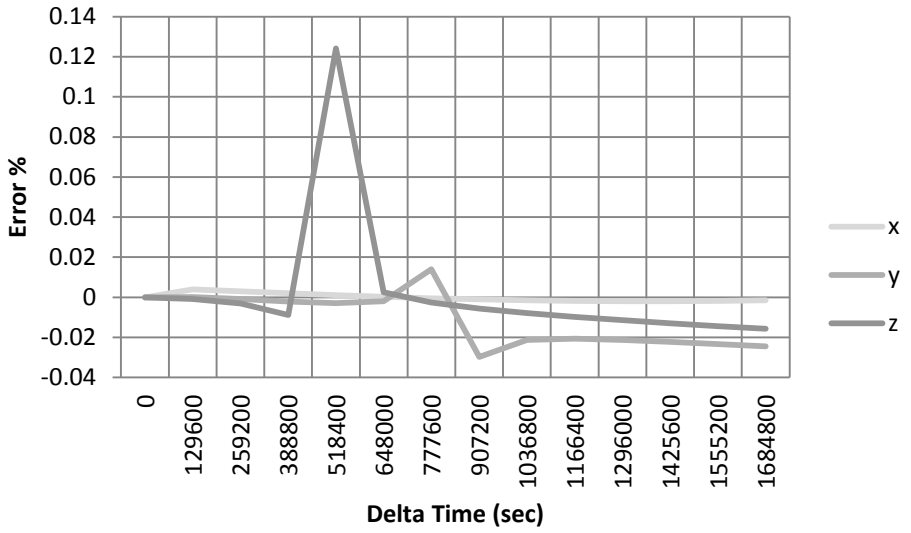

Figure 5-4. Inertial Position Vector Error Testing. Percentage error between inertial position vectors of STK and Bouquet.

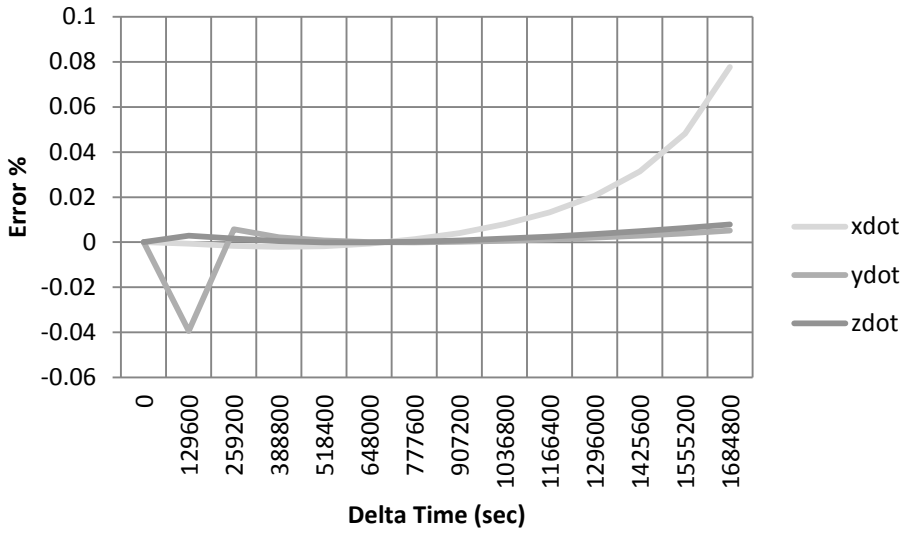

Figure 5-5. Inertial Velocity Vector Error Testing. Percentage error between inertial vectors of STK and Bouquet.

6. FUTURE WORK

6.1 Beta Release

The first release of Bouquet as a beta version will be done by middle of June 2011. Release as a beta allows multiple users to test the application for bugs and suggestions. Currently, a handful of necessary improvements are needed before the initial release.

These improvements are:

- A scenario save and load feature.
- Extending compatible relative trajectories to the correct number of periods.
- Display of relative trajectories to the 2D projection map.
- Constellation class and grouping of satellites.
- Accounting of J_2 drift effects.
- A preliminary user instruction manual.

6.2 Update and Support

Long-term goal of Bouquet is to release it to the public as an open source application with continual improvement and support, eventually offering a powerful satellite and constellation analysis tool that is an alternative to commercial programs.

Development can be expanded to a collaborative model between many programmers. For this long-term objective to be realized, several factors must be satisfied.

First, a continually operational server, with sufficient bandwidth, to store the source code and downloadable executable file is necessary. This can easily be achieved through use of the free Texas A&M University server space afforded to professors or through a private internet service provider if so desired.

Second, a collaboration framework must be in place to handle multiple developers making edits to the source code. To this end, the development of Bouquet already uses the open source Subversion tracking system which is capable of tracking and organizing changes made by multiple programmers [16]. A public forum space will also facilitate in communication between users, developers and core supporters.

Third, extensive documentation about the internal classes and code is necessary to allow future developers to easily gain the knowledge necessary to contribute. For Bouquet, it was decided that the open source Doxygen document generator would be used to make class references.

Finally, a group of core supporters should be designated to offer a general direction, set objectives, moderate edits, track bugs and reply to community concerns for Bouquet.

6.3 Planned Improvements

Any computer application will have an infinite set of desired improvements, thus it is important to find the most effective improvements and prioritize them. Immediately

useful and necessary improvements to Bouquet are:

- Implementation of the FC optimization methods.
- Multiple scenario handling system, such that users can load scenarios created by others and switch between different scenarios without needing to close and reopen the program.
- Redesign the 2D projection system to a smoother alternative that is able to handle non-cylindrical projections.
- Expand the predefined list of planets, orbit models and projection models available.
- Expand the FC creation options.

Long-term desired capabilities include, but not limited to:

- Development of a variety of sensor classes that can be attached to satellites.
- Capability of satellites to track their attitude.
- Addition of surface objects that can interact with satellites.
- Capability to define borders and features on the surface through a vector map, allowing for opportunities such as the calculation of GPS signals available over Texas.
- Improving accuracy by accounting for more minute orbital drift effects

such as lunar gravity and atmospheric drag.

6.4 Known Bugs

Table 6-1 includes the currently known bugs that haven't been fixed yet. None of these bugs are fatal to the major functionality of Bouquet, and instead are limited to exceptional cases.

Table 6-1: Known bugs.

Name	Class	Description	Cause
Wobbling Axis	FrameSetup	Non-orthogonal axis behavior during rotation about [1, 1, 1] axis.	Unknown
T0 and Tf Input	MainWindow	Ambiguity on whether T0 or Tf is input first during certain input sequences.	Known
Clipping Plane	OrbitView	Front and back clipping plane is too close to camera.	Known
Vista Crash	unknown	Bouquet crashes in Vista when the Aero theme is disabled.	Unknown
Vista Texture	OrbitView	Vista does not recognize some common texture image types.	Known
Choppy Planet	PlanetClass	Planet rotation animation gets choppy when Tcdelta becomes large.	Known
Day Night Lighting	PlanetClass, SunClass	Positioning of the day light corresponding to the date and time is off.	Unknown.

7. CONCLUSIONS

The development of Flower Constellations by Dr. Daniele Mortari, his colleagues and his students offer a powerful new method for analyzing groups of satellites, pushing the perspective a step further from the old Walker Constellation methods. Bouquet was developed to realize the full potential of this new theory.

With emphasize on creating a practical, useful and long lasting product, the initial development outlined in this thesis create the foundations of the tool that is flexible and upgradable. The effort to make sure that Bouquet is brought to its full potential through continuous support is highlighted by the focus on open source development and eye towards collaboration. The ultimate goal, and hope, of the team is for Bouquet to become a viable free alternative to commercial implementations such as STK, offering scientists, engineers, academics and students another path towards appreciating, understanding and utilizing the space frontier.

REFERENCES

[1] Mortari, D., Wilkins, M. P., Bruccoleri, C., "The Flower Constellations," *Journal of Astronautical Sciences*, Vol. 52, No. 1, 2004, pp. 107-127.

[2] Avendaño, M., Mortari, D., "Rotating Symmetries in Space – The Flower Constellations," *AAS Journal*, Vol. 189, No. 9, 2009, pp. 1317-1334.

[3] Walker, J. G., "Satellite Constellations," *Journal of the British Interplanetary Society*, Vol. 37, 1984, pp. 559-571.

[4] Avendaño, M., Mortari, D., Davis, J. J., "The Lattice Theory of Flower Constellations," *AAS Journal*, Vol. 172, No. 10, 2011 (to be published).

[5] STK, Satellite Tool Kit, Software Package, Ver. 8.1.3, Analytical Graphics, Inc., Exton, PA, 2008, URL: http://www.agi.com/products/by-product-type/applications/stk/.

[6] GMAT, General Mission Analysis Tool, Software Package, Ver. 12.12.07.12.53.42, NASA Goddard Space Flight Center, Greenbelt, MD, 2007, URL: http://gmat.gsfc.nasa.gov.

[7] Google Earth, Software Package, Ver. 6.0.3.2197, Google, Mountain View, CA, 2011, URL: http://www.google.com/intl/en/earth/index.html.

[8] Celestia, Software Package, Ver. 1.6.0, Laurel, C., 2009, URL: http://www.shatters.net/celestia/.

[9] Wright Jr., R. S., Lipchak, B., Haemel, N., *OpenGL SuperBible*, 4th ed., Addison-Wesley, Boston, 2007.

[10] VS2008, Microsoft Visual Studio 2008 Professional Edition, Software Package, Ver. 9.0.30729.1 SP, Microsoft, Redmond, WA, 2007.

[11] Khronos Group, "Windows Vista and OpenGL-the Facts," *The OpenGL Pipeline Newsletter*, Vol. 3, URL: http://www.opengl.org/pipeline/article/vol003_9/ [cited 2 May 2011].

[12] Feldman, A., Daymon, M., *WPF in Action with Visual Studio 2008*, Manning Publications, Greenwich, CT, 2009.

[13] Blanchette, J., Summerfield, M., *C++ GUI Programming with Qt 4*, 2nd ed., Prentice Hall, Boston, 2008.

[14] Qt Framework, Software Package, Ver. 4.7.0, Nokia, Espoo, Finland, 2010, URL: http://qt.nokia.com/.

[15] Malik, D.S., *C++ Programming: Program Design Including Data Structures,* 2nd ed., Thomson Course Technology, Boston, 2004.

[16] Collins-Sussman, B., Fitzpatrick, B. W., Pilato, C. M., *Version Control with Subversion* [electronic book], O'Reilly Media, Sebastopol, CA, 2008, URL: http://svnbook.red-bean.com/ [cited 16 April 2011].

APPENDIX A

Table A-1. Satellite 1 details.

Satellite 1, Critically Inclined Orbit	
Semi-major Axis (km)	14628.137
Eccentricity	0.461439
Inclination (deg)	63.435
RAAN (deg)	351.437
Argument of Perigee (deg)	270
Initial True Anomaly (deg)	82.552
Initial Mean Anomaly (deg)	34.158
Period (sec)	17607.4
Initial Date (UTC)	June 7, 2011 1300
Final Date (UTC)	June 7, 2011 1800
Output Time Interval (min)	30

Table A-2. Satellite 2 details.

Satellite 2, Molniya Orbit	
Initial Pos. X (km)	-513.826
Initial Pos. Y (km)	-3036.582
Initial Pos. Z (km)	-6150.115
Initial Vel. X (km/s)	9.90373
Initial Vel. Y (km/s)	-1.67583
Initial Vel. Z (km/s)	0
Period (sec)	43061.6

Initial Date (UTC)	June 7, 2011 1300
Final Date (UTC)	June 27, 2011 1300
Output Time Interval (day)	1.5

Table A-3. Comparison of X coordinate position between STK and Bouquet.

Time			(UTC)	x (km)		
				STK	Bouquet	Error %
7	June	2011	1300	-513.826227	-513.826	4.41784E-07
8	June	2011	0100	3514.409966	3500.5	0.00395798
10	June	2011	1300	6997.003584	6975.73	0.003040385
11	June	2011	0100	9763.150149	9742.61	0.002103844
13	June	2011	1300	11919.53022	11905.6	0.001168688
14	June	2011	0100	13608.43504	13604.3	0.000303859
16	June	2011	1300	14942.28601	14948.9	-0.000442635
17	June	2011	0100	16001.8067	16018.8	-0.001061962
19	June	2011	1300	16844.52108	16870.2	-0.001524467
20	June	2011	0100	17512.10964	17544.4	-0.001843888
22	June	2011	1300	18035.46691	18071.6	-0.002003446
23	June	2011	0100	18438.02352	18475.1	-0.00201087
25	June	2011	1300	18737.93111	18772.7	-0.001855535
26	June	2011	0100	18949.52309	18978.6	-0.00153444

Table A-4. Comparison of Y coordinate position between STK and Bouquet.

Time			(UTC)	y (km)		
				STK	Bouquet	Error %

7	June	2011	1300	-3036.582372	-3036.58	7.81141E-07
8	June	2011	0100	-3405.686281	-3405.49	5.76333E-05
10	June	2011	1300	-3209.480188	-3212.24	-0.000859894
11	June	2011	0100	-2668.009865	-2673.48	-0.002050268
13	June	2011	1300	-1952.892051	-1958.49	-0.002866492
14	June	2011	0100	-1158.030644	-1160.3	-0.001959668
16	June	2011	1300	-331.38399	-326.729	0.014047118
17	June	2011	0100	502.210185	517.161	-0.029770035
19	June	2011	1300	1329.514923	1357.76	-0.021244648
20	June	2011	0100	2143.325398	2187.51	-0.020614976
22	June	2011	1300	2939.704029	3002.1	-0.021225256
23	June	2011	0100	3716.547922	3799.08	-0.02220665
25	June	2011	1300	4472.814151	4577.07	-0.023308782
26	June	2011	0100	5208.084358	5335.36	-0.024438091

Table A-5. Comparison of Z coordinate position between STK and Bouquet.

	Time	(UTCG)	z (km)		
			STK	Bouquet	Error %
7	June	2011	1300	-6150.11534	-6150.12	-7.57709E-07
8	June	2011	0100	-5534.770527	-5539.01	-0.000765971
10	June	2011	1300	-3988.143509	-4000.66	-0.003138425
11	June	2011	0100	-2000.403923	-2018.02	-0.00880626
13	June	2011	1300	126.083999	110.429	0.124163249
14	June	2011	0100	2253.834354	2247.98	0.002597509
16	June	2011	1300	4325.874571	4337.26	-0.002631937

17	June	2011	0100	6320.194919	6355.28	-0.005551266
19	June	2011	1300	8229.897704	8294.07	-0.00779746
20	June	2011	0100	10054.68367	10152.4	-0.009718489
22	June	2011	1300	11797.09307	11932	-0.011435608
23	June	2011	0100	13460.79181	13635.6	-0.012986472
25	June	2011	1300	15049.77403	15266.6	-0.014407257
26	June	2011	0100	16567.99152	16828.3	-0.015711529

Table A-6. Comparison of X coordinate velocity between STK and Bouquet.

	Time	(UTCG)	vx (km/sec)		
			STK	Bouquet	Error %
7	June	2011	1300	9.90373199	9.90373	2.00934E-07
8	June	2011	0100	9.216513547	9.22175	-0.00056816
10	June	2011	1300	7.511277497	7.5234	-0.001613907
11	June	2011	0100	5.869418333	5.88183	-0.002114633
13	June	2011	1300	4.579669902	4.58787	-0.001790543
14	June	2011	0100	3.603024954	3.60539	-0.000656406
16	June	2011	1300	2.855952415	2.85229	0.00128238
17	June	2011	0100	2.271978568	2.26268	0.004092718
19	June	2011	1300	1.805042134	1.79067	0.007962215
20	June	2011	0100	1.423901118	1.40502	0.013260133
22	June	2011	1300	1.107134376	1.08428	0.02064282
23	June	2011	0100	0.839751088	0.813408	0.031370115
25	June	2011	1300	0.611021482	0.58161	0.048134939
26	June	2011	0100	0.413095712	0.380988	0.077724632

Table A-7. Comparison of Y coordinate velocity between STK and Bouquet.

		Time	(UTCG)	vy (km/sec)		
				STK	Bouquet	Error %
7	June	2011	1300	-1.675830461	-1.67583	2.75087E-07
8	June	2011	0100	-0.129173923	-0.134257	-0.039350644
10	June	2011	1300	0.975744435	0.970173	0.005709933
11	June	2011	0100	1.564548058	1.56115	0.00217191
13	June	2011	1300	1.844953809	1.84361	0.00072837
14	June	2011	0100	1.966586917	1.96638	0.000105216
16	June	2011	1300	2.006932988	2.00701	-3.8373E-05
17	June	2011	0100	2.004498305	2.0042	0.000148818
19	June	2011	1300	1.978946978	1.9778	0.00057959
20	June	2011	0100	1.940740879	1.9384	0.001206178
22	June	2011	1300	1.895666531	1.8919	0.001986916
23	June	2011	0100	1.847028709	1.84166	0.002906673
25	June	2011	1300	1.796760217	1.78966	0.003951678
26	June	2011	0100	1.746008107	1.73708	0.00511344

Table A-8. Comparison of Z coordinate velocity between STK and Bouquet.

		Time	(UTCG)	vz (km/sec)		
				STK	Bouquet	Error %
7	June	2011	1300	0	0	0
8	June	2011	0100	2.816349427	2.80808	0.002936222
10	June	2011	1300	4.423756694	4.41682	0.001568055
11	June	2011	0100	5.036067849	5.03351	0.000507906

13	June	2011	1300	5.158468163	5.15854	-1.3926E-05
14	June	2011	0100	5.072568244	5.07295	-7.52589E-05
16	June	2011	1300	4.903104243	4.90204	0.000217055
17	June	2011	0100	4.70374645	4.70006	0.000783726
19	June	2011	1300	4.497866446	4.49081	0.001568843
20	June	2011	0100	4.295654317	4.28476	0.002536125
22	June	2011	1300	4.101366583	4.08634	0.003663799
23	June	2011	0100	3.916515699	3.89717	0.004939518
25	June	2011	1300	3.741332427	3.71755	0.006356673
26	June	2011	0100	3.575459918	3.54718	0.007909449

VITA

Name:	Mandakh Enkh
Address:	Department of Aerospace Engineering c/o Dr. Daniele Mortari Texas A&M University College Station, TX 77843-3141
Email Address:	kennyavir@gmail.com
Education:	B.S., Aerospace Engineering, Texas A&M University, 2008 M.S., Aerospace Engineering, Texas A&M University, 2011